8 Conclusion

The trend towards larger DRAM devices exacerbates the processor / memory bottleneck, requireing costly cache hierarchies to effectively support high performance microprocessors. A viable alternative is to move the processor closer to the memory, by integrating it onto the DRAM chip. Processor / memory integration is advantageious, even if it requires the use of a simpler processor.

It was shown that a conventional, single scalar processor with a small cache, integrated with a 256 Mbit DRAM array can form a selfcontained, general purpose processing element with comptetitive performance that can approach that of high-end superscaler processors with large, multilevel caches. Small (8k) direct mapped instruction caches with long lines (512 bytes) perform surprizingly well with a zero-fill cost, which is a featue of the integration. Victim caches were shown to be very effective when combined with a multi-banked column buffer data cache. Combined with the lower latency inherent in an integrated design, the memory induced degradation of processor performnace was gretly reduced.

Including scalable distributed shared memory (DSM) with hardware assisted cache coherency, results in a versatile builing block that can outperform an ideal (= infinite second level cache) convential DSM system.

References

[1] Wulf,William.A and McKee,Sally A. *Hitting the Memory Wall: Implications of the Obvious.* ACM Computer Architecture News. Vol.23, No.1 March 1995.

[2] Nowatzyk, A., Aybay, G., Browne, M., Kelly, E., Parkin, M., Radke, B. and Vishin, S. *The S3.mp Scalable Shared Memory Multiprocessor.* Proc. of the 24th Int'l Conf. on Parallel Processing, 1995.

[3] Nowatzyk, A., Browne, M., Kelly, E. and Parkin, M. *S-Connect: from Network of Workstations to Supercomputer Performance*. Proc. of the 22nd Int'l Symp. on Computer Architecture, June 1994.

[4] Nowatzyk, A., Aybay, G., Browne, M., Kelly, E., Parkin, M., Radke, B. and Vishin, S.*Exploiting Parallism in Cache Coherency Protocol engines*, Europar 1995, Stockholm, Sweden

[5] Rambus System Specification, 1993, Rambus Inc., Mountain View, CA

[6] SDRAM: *1994 Samsung SDRAM Databook*, Samsung Semiconductor Inc., 3655 N. first St., San Jose CA 95134,

[7] H. Koike, et.al., *A 30ns 64Mb DRAM with Built-in Self-Test and Repair Function*,ISSCC 1994, San Francisco

[8] Wilkes, Maurice.V., The Memory Wall and the CMOS End-Point, ACM Computer Architecture News. Vol. 23, No. 4 September 1995.

[9] Lenoski, D. *The Design and Analysis of DASH: A Scalable Directory-Based Multiprocessor.* PhD Dissertation, Stanford University, December 1991.

[10]Agarwal, A.,Kubiatowicz J., Kranz, D., Lim, B., Yeung, D., D'Souza, G., Parkin, M. *Sparcle: An Evolutionary Processor Design for Large-Scale Multiprocessors*. IEEE Micro, June 1993, pages 48-61.

[11]Jouppi, Norm. *Improving Direct-Mapped Cache Performance by Addition of a Small Fully-Associative Cache and Prefetch Buffer*, Proceedings of the 17th Annual International Symposium on Computer Architecture, 1990 pages 364-373

[12]The SPEC 95 benchmark suite, Spec International

[13]SHADE simulator, SunLabs Technical Report, 1993

[14]Our SPEC 95 simulation technical report

[15]Inmos Transputer Family databook.

[16]Ref for 1Gbit and 256Mbit DRAMS, NEC personal communication.

[17]Brorsson, M., Dahlgren, F., Nilsson, H. and Stenstrom, P. *The CacheMire Test Bench - A Flexible and Effective Approach for Simulation of Multiprocessors*. Proc. of the 26th Annual Simulation Symp., pp. 115-124, 1993.

[18]Dubois, M., Skeppstedt, J., Ricciulli, L., Ramamurthy, K. and Stenstrom, P. *The Detection and Elimination of Useless Misses in Multiprocessors*. Proc. of the 20th Annual Int'l Symp. on Computer Architecture, pp. 88-97, May 1993.

[19]Singh, J.P., Weber, W.-D., and Gupta, A. *SPLASH: Stanford Parallel Applications for Shared-Memory.* Computer Architecture News, 20(1):5-44, March 1992.

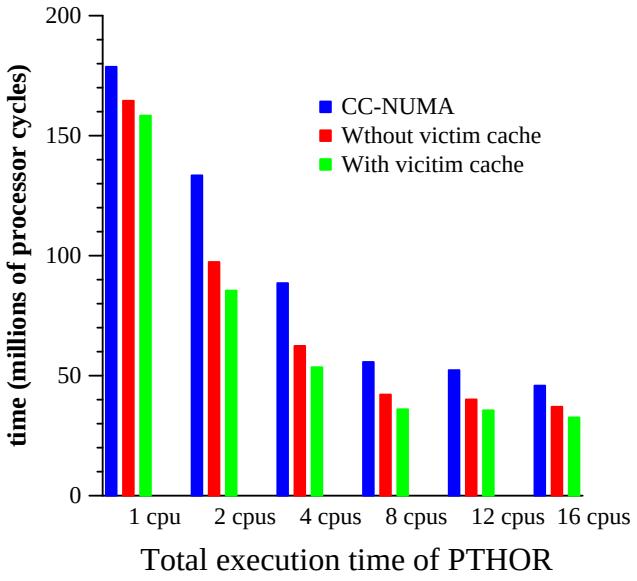
[20]Marsan,G,Conti, *A class of generalized stochastic ptrinets for the performance evaluation of multiprocessor systems*, ACM Transactions on Computer Systems, 2(2) : 93, May 1984

[21]Dally, William.J. et. al. *M-Machine Microarchitecture*, Tech Report, Artificial Intelligence Lab MIT, Cambridge, MA. Jan 1993

[22]Kogge, Peter.M., *EXECUBE - A New Architecture for Scalable MPPs*, 1994 Intl. Conf. on Parallel Proc.

[23]*ADSP-21060 SHARC Super Harvard Architecture Computer*, ANALOG DEVICES, Norwood, MA, Oct. 1993.

FIGURE 17 : PTHOR



6 Related Work

Transputer

In 1982 Inmos started a family of RISC processor chips [15] which were combined with a small amount of local memory and tightly integrated with a 4 serial link communications interface. The processor was tailored specifically for a message passing paradigm (specifically the language Occam), and efficiently supported high speed context switching and message handling. While this chip has some similarities with the idea presented it did not provide any kind of shared memory capability or even virtual memory. Moreover the local memory was simply 1KB of static RAM, so for all practical applications the chip had to be combined with external DRAM.

M machine

The M-Machine [21] project also plans to integrate 4 superscalar cpus with memory on a single chip. However, the on-chip memory size is small as 4M bits. In most cases, external memory and I/O chips are needed. Moreover, the M-Machine is a message massing design. On the other hand, our vision is to build efficient and scalable shared-memory system on top of advanced DRAM technology which integrates one cpu, large memory and communication links on a single chip.

Exacube

The EXECUBE by Kogge [22] exploited a similar idea of utilizing the memory bandwidth to the maximum by cpu memory integration. A cpu element, 64KB memory and four communication links form a processing unit. Each chip based on the dense DRAM technology of the day holds eight such processing units. Unlike our design, the EXECUBE chip is intended to serve as building block for MPP systems. There is no shared memory capability, or ability to use DRAM column buffers for I & D caches. Furthermore it is primarily intended as a SIMD architecture.

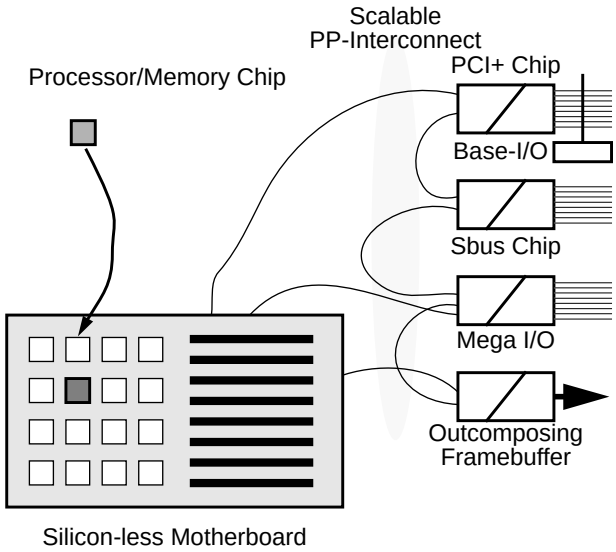
Sharc

Another design integrates the cpu with memory is the Analog Devices ADSP-21060 chip [23]. Each chip has a single DSP (digital signal processor) cpu with 4M bits. Like the EXECUBE, MPPs systems can also be built using this chip, but not shared-memory systems.

7 Vision

The integrated processor/memory chip could be regarded as the equivalent to a Lego(tm) building block. Dwafted by its modest heat-sink to cool some 1.5W, the processing elements could be added incrementally to a silicon-less mother board (Figure 20) that provides only sockets with power and signal traces to other such sockets. As needed, P/M chips are added to a logically seamless shared memory multiprocessor.

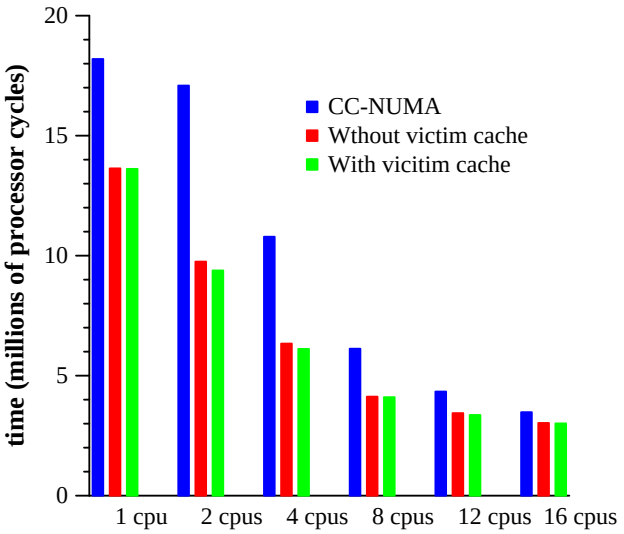
FIGURE 18 : Silicon-Less Motherboard



I/O functions are handled by chips that can translate to standard interfaces, such as PCI, SCSI, ATM, Fiberchannel, etc. Among the more interesting capabilities of such a system is to build a framebuffer that retrieves its data from the main memory as it refreshes the screen or LCD panel. This is made feasible by the high memory bandwidth that is available internally.

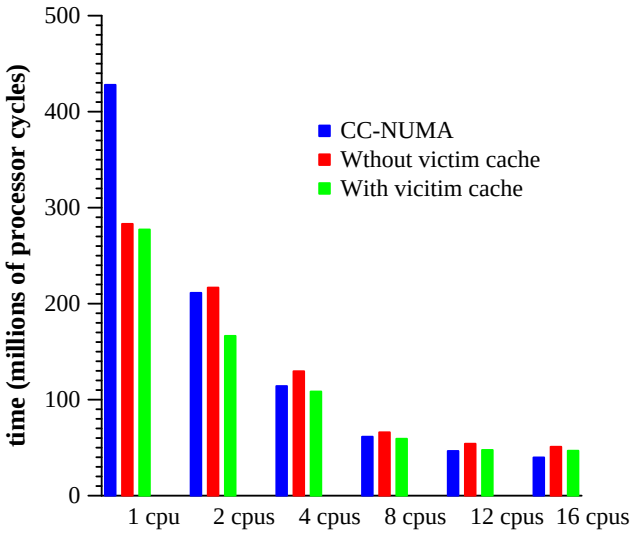
Due to the scalabe interconnect fabric, the system bi-section bandwidth increases as components are added. This system is not limited to one mother-board, rather the delay insensitive point-to-point connections of the S-Connect systems allow multiple boards to communicate via cables (< 10 m) or fiber-optic connections (up to 200m).

FIGURE 14 : MP3d



Total execution time of MP3d

FIGURE 15 : OCEAN

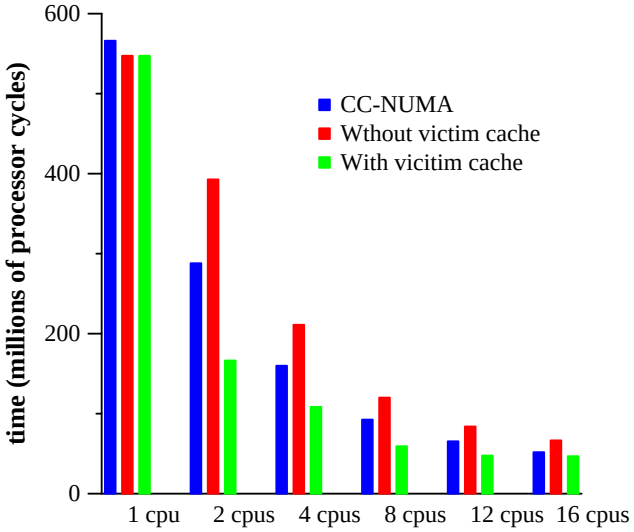


Total execution time of OCEAN

A second observation is that the column buffer D-cache itself is not enough to support scalable computation. As shown in figures 17 and 18, the reference CC-NUMA is better than the integrated design with only column buffers. Again this is because the addition of coherence traffic from shared memory operation further increases the conflict misses caused by the small number of column buffer long lines. As a result, in our simulations most memory accesses miss the D-cache and need 6 cycles to reach the data in the INC DRAM in the integrated design; while the similar accesses may be served in 1 cycle by FLC for the reference CC-NUMA design. This is not quite a fair comparison due to the small nature of the standard problem's working sets - for larger problems the FLC of the reference CC-NUMA architecture would start experiencing more capacity misses incurring the 6 cycle penalty to it's SLC.

Based on the preliminary observations, the integrated design was augmented just as for the uniprocessor case by the addition of a small 16 way by 32B associative victim cache per node. Compared to the 16K FLC of the reference CC-NUMA, the 512B capacity of the victim cache is very small however, even this small victim cache can reduce the execution time of the integrated design by half in the case of WATER. With the aid of victim caches, the integrated design typically shows the best performance results except for the WATER application.

FIGURE 16 : WATER



Total execution time of WATER

The simulation benchmarks are taken from the SPLASH suite [19]; descriptions and sizes of the data sets are given in table 3.

TABLE 5 : Splash Benchmarks

Benchmark	Description	Data Set
LU	LU decomposition	200x200 matrix
MP3d	3-D particle-based wind-tunnel simulator	10 K particles, 10 steps
Ocean	Ocean basin simulator	128x128 grids, tolerance 10^{-7}
Water	N-body water molecular dynamics simulation	288 molecules, 4 time steps
PTHOR	Distributed digital time digital circuit simulator	RISC circuit, 1000 time steps

The architectural simulator is built on top of the CacheMire Test Bench [17], which is an execution-driven functional simulator of multiple SPARC processors. The processors in the CacheMire simulator issue memory accesses and our architectural simulator then delays the processors according to the timing models as depicted in table 4. Since the SPLASH application codes are so small we only simulate data references for both the integrated and the reference architectures; namely, instruction fetches are assumed to always hit in the respective level 1 instruction caches.

TABLE 6 : Memory Latency (processor cycles)

System	Access	Latency
	Hit in column buffer	1
	Hit in victim cache	1
	Access local memory & INC	6
	Invalidation round trip delay	80
	Load remote data	80
Comparative CC-NUMA	Hit in FLC	1
	Hit in SLC	6
	Invalidation round trip delay	80
	Load remote data	80

5.2 Performance Results

The total execution times for the benchmarks LU, MP3d, OCEAN, WATER and PTHOR are depicted in figures 15 through 19 for increasing numbers of processors.

The following observations can be made: Firstly, the column buffers which load 512 bytes at a time exploit the spatial locality of local memory accesses well with their prefetching effect. As a result, our design outperforms the traditional CC-NUMA designs for small numbers of processors in all cases - until the working set per cpu becomes too small for the effect to be pronounced.

It should be noticed though that the coherence units transferred between nodes are always 32 byte blocks. Therefore, the traditional CC-NUMA design must count the same number of cold and coherence SLC misses as the cold and coherence INC misses in our design. The long line prefetching effect does not help for accesses to data held on another node. This is acceptable because the false-sharing costs of long lines would outweigh the prefetching benefits for most applications. However, the long column buffer lines can improve the performance of the integrated multiprocessor's remote data caches. The long column buffer lines enable access to seven 32 bytes INC blocks each - providing 7 way associativity for cached remote memory reducing conflict misses.

Therefore, when applications exploit good sequential locality for local memory accesses or a large number of conflicts with remote data accesses, our design with column buffers can produce good results.

WATER is the only benchmark for which the reference CC-NUMA design shows better results than the integrated architecture unaided by the victim caches. In WATER, the main data structure is a shared vector which maintains all molecules. Sets of molecules are allocated to processors statically. In this application, true sharing misses dominate. As each molecule is described by a data structure of approximately 600 bytes which is only partially accessed, the limited numbers column buffers of 512 bytes suffer from the lack of spatial locality. As shown in figure 18, when the victim caches are used, the total execution time is significantly reduced.

FIGURE 13 : LU

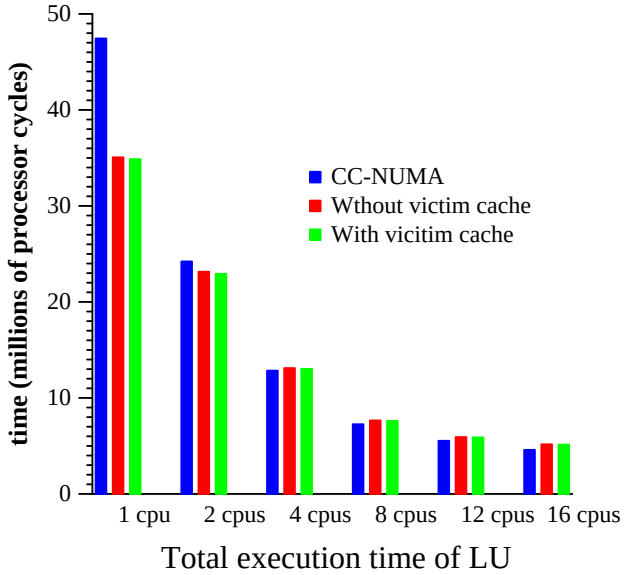
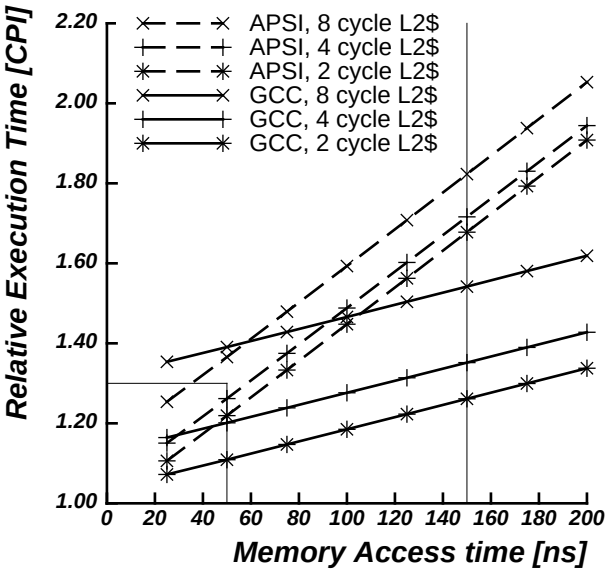


FIGURE 12 : Cache Latency Impact (??)



Based on the Shade simulations, the miss-rates and load store ratios were used to define the parameters for the GSPN model. In the case of the floating point benchmarks, the CPI figure was multiplied by the CPI rate of the MicroSparc-II processor for an ideal memory to account for the fact that not all floating point operation will complete in one cycle

.

TABLE 3 : Spec'95 Estimates, no Victim Cache

Nmae	IPC	Spec-ratio
124.m88ksim	0.91	5.64
126.gcc	0.87	6.93
129.compress	0.88	
130.li	0.93	
132.ijpeg	0.98	5.48
134.perl	0.77	6.43
147.vortex	0.84	
101.tomcatv	0.64	12.85
102.swim	0.52	16.82
103.su2cor	0.68	6.28
104.hydro2d	0.95	9.59
107.mgrid	0.99	5.41
110.applu	0.98	7.65
125.turb3d	0.92	7.24
141.apsi	0.91	12.82
145.fpppp	0.93	
146.wave5	0.73	14.42

.

TABLE 4 : Spec'95 Estimates, withVictim Cache

Nmae	IPC	Spec-ratio	DEC 21164
124.m88ksim	0.93	5.70	7.2
126.gcc	0.88	7.03	6.4
129.compress	0.90		6.7
130.li	0.95		6.6
132.ijpeg	0.99	5.51	6.9
134.perl	0.81	6.71	7.9
147.vortex	0.86		7.4

TABLE 4 : Spec'95 Estimates, withVictim Cache

Nmae	IPC	Spec-ratio	DEC 21164
101.tomcatv	0.86	17.38	14.7
102.swim	0.88	28.50	19.2
103.su2cor			6.9
104.hydro2d	0.97	9.78	7.8
107.mgrid	0.99	5.42	9.3
110.applu	0.98	7.65	6.5
125.turb3d	0.94	7.35	11.0
141.apsi	0.93	13.1	13.6
145.fpppp	0.94	14.71	21.4
146.wave5	0.88	17.30	14.7

5 Multiprocessor performance

Aside from the overall design we have only considered the uniprocessor performance aspects of our integrated system. The unusual design of the processor caches was shown in most cases to be beneficial, but is not clear that this should also be the case for shared memory multiprocessing where the addition of coherence misses affects accesses to and through the caches.

5.1 Simulation Models and Methodology

To understand the performance potential of our integrated design, we use execution driven simulation with a timing accurate architecture model to evaluate a simple system based on the building block of Figure 3. These performance figures are then compared with results obtained from a reference CC-NUMA design [9]] with 16K direct-mapped FLCs and infinitely sized SLCs. The use of infinite SLCs removes the capacity misses of the SLCs so that it represents an ideal upper-bound of the system performance when only the cold and the coherence misses are considered [18]. This enables the direct study of the integrated cache system compared to a conventional first level cache in a shared memory system. In addition we selectively enable and disable the victim caches for the integrated system to study their performance merits.

To simulate the proposed integrated system, the inter-node cache[1] (INC) size was set at one megabyte per memory/processor node - larger than the working sets of the applications used and so comparable to the infinite SLCs of the reference CC-NUMA architecture. However, a finite size was chosen because the INCs are configured as 7-way set-associative caches and therefore making them infinitely large would remove the beneficial effects of set associativity. The granularity of coherence or, "coherence unit" size, is always 32 bytes. For the purposes of these simulations cache coherence was maintained by a write-invalidate protocol [18].

1. A dedicated area of memory functioning as a remote data cache for a CC-NUMA style shared memory architecture.

FIGURE 10 : Processor/Cache GSPN

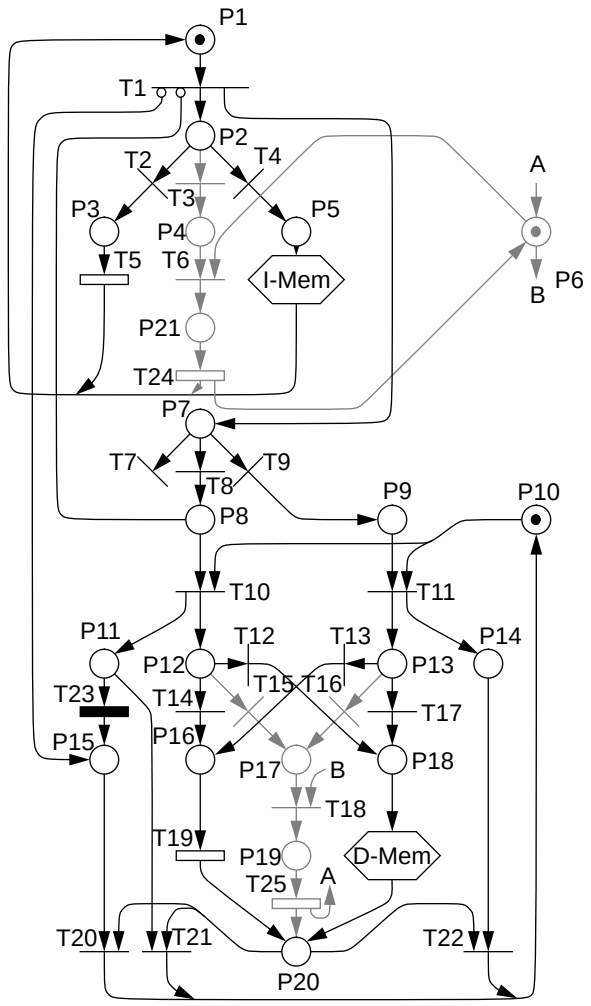


Figure 10 depicts both of the complete processor models. The additional components required to model the shared second level cache of the conventional “reference” system architecture are shown in grey. These components are removed for simulating our proposed “integrated” system.

The top part of Figure 10 is the instruction fetch unit. Each time the transition T1 fires, an instruction is executed. T1 depends on a loaded instruction (P1) which is retrieved from either the instruction cache (P3/T5), the optional second level cache (P21/T24) or the memory (P5...). The rates on the transitions T2, T3, and T4 reflect the probabilities for an instruction cache hit, a second level cache hit, or a fill from main memory respectively. These rates were determined from the Shade simulations of the actual benchmarks.

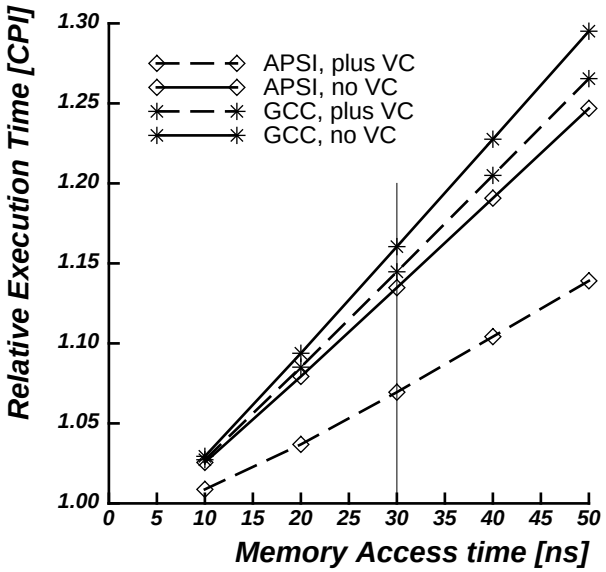
Each time T1 fires, an instruction is presented to the execution units. The transitions emanating from place P5 reflect the probability for an non-Load/Store operation (T7), a Load (T8) or a Store (T9). Operations other than loads or stores are assumed not to cause pipeline stalls. This is a simplification in regard to floating point operations that may occupy a functional unit for more than one cycle.

A store buffer is assumed that can postpone stores (hence P9 does not stall the processor). However, the load/store unit is limited to **one** outstanding operation at a time (the token from P10 is require to issue either a load or a store). The exponentially timed transition T23 models stalls due incomplete loads. To model a system without scoreboarding, this rate for T23 is set to infinity. The integrated system assumes the presence of scoreboarding logic, but the rate of T23 to is set 1 so that on average on one operation can be performed before an incomplete load will stall the processor.

The transition ratios for T14, T15 and T12 reflects the probabilities that a load is a data cache hit, a second level cache hit or a reference to main memory. Similarly, T13, T16, and T17 model the cache behavior of stores. All of these rates are determined from the Shade cache simulation of the actual benchmarks.

The second level cache uses the grey GSPN components. The deterministically timed transitions T24 and T25 model the access times. The place P6 serves to ensures mutual exclusion between data and instruction fetches.

FIGURE 11 : Memory Latency Impact



ture that saved the most recently accessed 32 byte subblock of columns that are replaced.

The buffer is the same size as a single cache line, though it is organized as a 16 way fully associative bank of 32B lines. When a cache line is to be evicted from a DRAM column buffer due to a miss, the last 32B sub-section of the main cache line being replaced is copied into a selected entry in the victim cache before the new cache line is read. Please note that this has no performance penalty, since the cpu is frozen on the miss, and the new cache line cannot be loaded until the access of the DRAM array is complete - leaving the system four free cycles to copy the 32byte block into the victim cache 64bits at a time.

On each memory access cycle, as the main D-cache is searched, each of the 16 entries in the victim cache are also check to see if once holds the correct 32B sub-section of the data to be accessed. Unlike a conventional victim cache, the contents cannot be reloaded into a line of the main D-cache because of the size disparity.

The addition of the Victim Cache was simulated and the resultant miss rates collected and presented in the far right column of the results for each application in figure 8.

The replacement algorithm used for the slots in the Victim Cache was *not-last-used* - essentially random.

The result of adding the Victim cache was dramatic In all but one application the combined D-cache and victim cache had a lower miss rate than the 16KB direct mapped data cache. For the applications where the D-cache had suffered on its own - particularly 103.su2cor and 101.tomcatv - the victim cache absorbed the conflict misses to reduce the miss rate to approximately that of a conventional 2 way 16KB cache. Moreover for three other applications, 102.swim, 146.wave5, 130.li, the miss rate was reduced between two and five fold - to a level equivalent to a 64KB conventional cache. Notice that for 102.swim and 146.wave5 the 2 way associative conventional 16K D-cache has a slightly higher miss rate than the direct mapped one. It would appear that for these applications a high degree of associativity is need when the D-cache is small - something which the victim cache provides, absorbing the accesses with poor spatial locality leaving the remaining accesses to benefit from the prefetching effect of the longer cache lines, and hence the miss rates almost as low as for the 64K conventional caches.

Sadly, there is always one exception, and while the victim cache helps reduce the miss rate by 25%, it does not have the capacity to absorb the conflicts from the main D-cache whose long lines are not suited to the poor spatial locality and small data structures used by the Go search algorithm

In summary; the long lines of the column buffer based D-cache provide a beneficial prefetching effect for some applications, but also cause significant cache conflicts for others due to the reduced total of lines in the cache. A modest victim cache the size of a single column buffer can absorb nearly all of the poor locality misses and in some cases allowing the prefetching effect to shine through.

4.5 Modeling Memory Contention

Although the column buffers comprising the data and instruction caches of the proposed design are filled in parallel and in a single cycle (after the DRAM array access and pre-charge times). Simply measuring the hit and miss ratios of the caches does not provide any information about bottlenecks and contention within the design which affect the throughput. Moreover it does not provide an entirely fair comparison with conventional processor designs which suffer even more from similar second-order contention effects arising from memory system bottlenecks. To examine this arena the integrated processor / memory system was modeled using generalized, stochastic Petri Nets (GSPNs) [20] that take into account contention for shared resources (such as memory banks) and event-dependencies.

The developed Petri nets were then adapted to also allow the modelling of a conventional architecture design with Harvard architecture first level instruction and data caches accessing a larger unified second level cache, which in turn would access a dual banked main memory.

Instruction and hit and miss ratios derived directly from the simulations described in the preceeding sections could then be dialed directly into the models.

The GSPNs were evaluated via Monte-Carlo simulations.

FIGURE 9 : Memory Bank GSPN

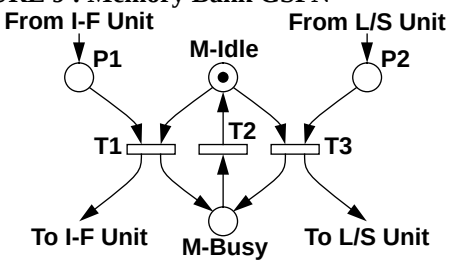


Figure 9 depicts the model for one memory bank that can either serve an instruction cache miss or a data cache miss, but not both simultaneously. The deterministically timed transitions T1 and T3 reflect the access time, while the transition T2 models pre-charge operations that prevent the memory from accepting a new transaction for a certain time after the last access. The places P1 and P2 are entered via immediate transitions from the instruction fetch unit or the load/store unit. These transitions have equal rates, so that memory banks are accessed with a uniform, random distribution. For the integrated processor/memory system, 16 memory banks are used. The conventional reference system which is used to validate the model has 2 independent memory banks.

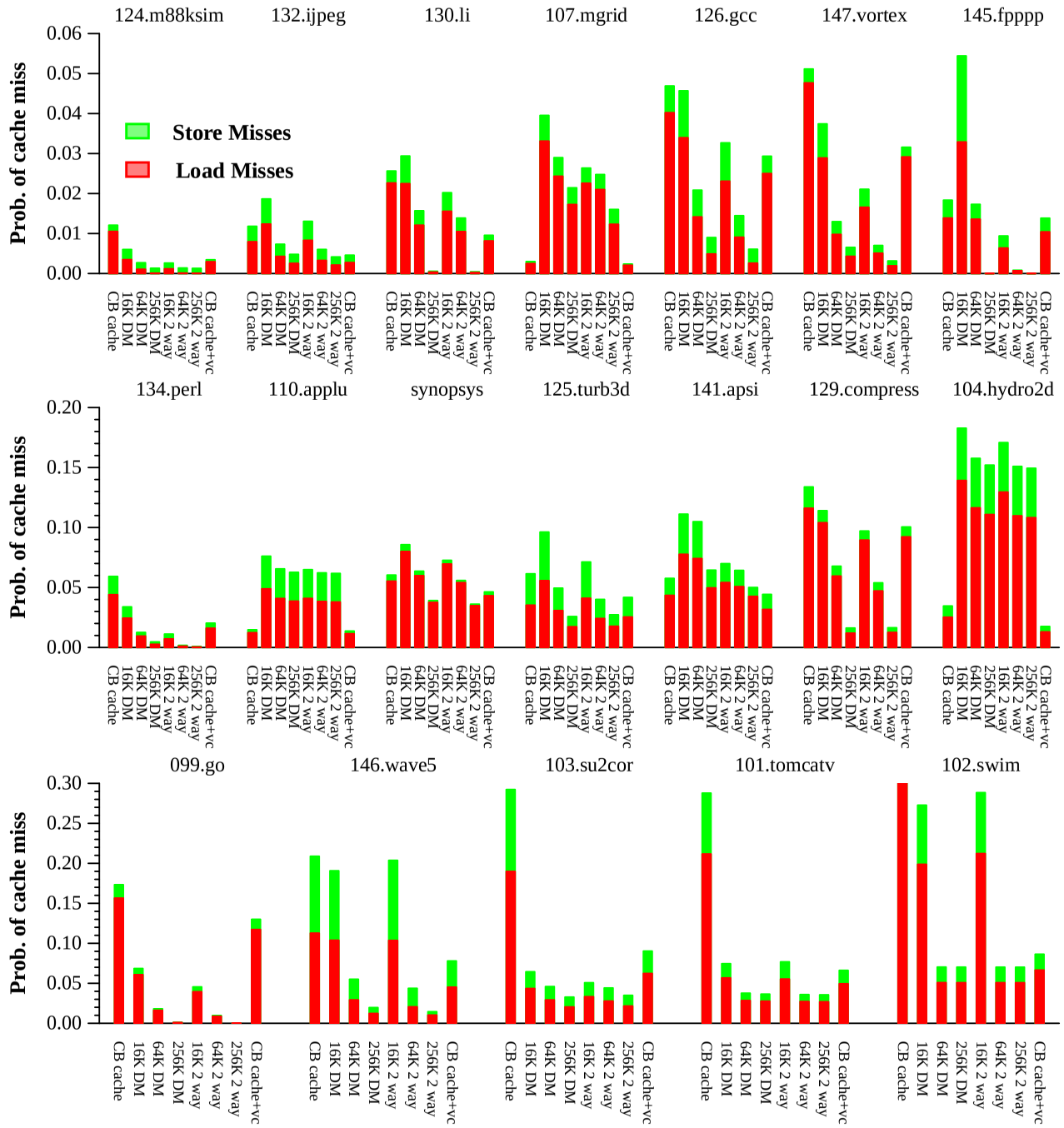


FIGURE 8 : Data Cache Miss Rates

Early design simulations, (not shown for space reasons), gave unacceptable miss rates for only an 8KB direct mapped cache with 512B lines - partly due to the reduced capacity but mostly due the conflicts arising from having only 16 cache lines. Introducing an additional data column buffer to each D-ram cell doubled the capacity of the architecture's D-cache to 16KB as mentioned, and provided two way associativity which dramatically improved the performance. While the prefetching benefits of the large D-cache lines are desirable, as can been seen from the miss rates in figure 8, the conflict misses caused by the long line size can be equally detrimental for other applications. It is necessary therefore to reduce the incidences of conflict without reducing the 512B line size or increasing the cache capacity.

4.4 Adding a Victim Cache

In [11] Norm Jouppi showed that a small full associative buffer could be used to hold cache lines most recently evicted from the data cache. This buffer worked effectively, in cases where the cache miss rate was dominated by conflicts, to increase the effective associativity of the cache.

To reduce the dominating effect of cache conflicts seen in section 4.3 for certain of the benchmark programs, a small associative buffer was added to the proposed architec-

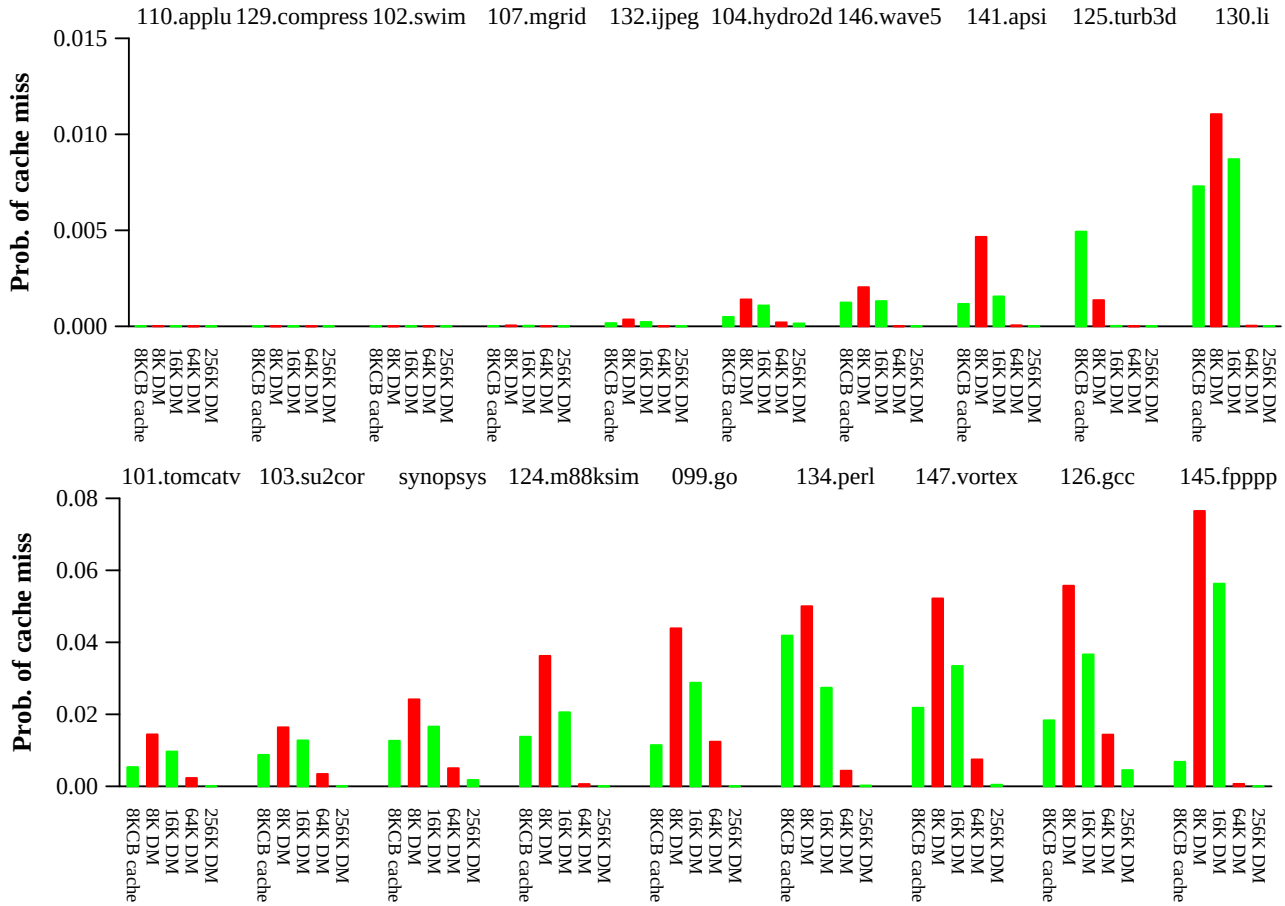


FIGURE 7 : Instruction Cache Miss Rates

4.3 Data Cache Performance

Instruction caches are important to keep the processor effectively utilized, and the generally good locality of instruction streams means that the prefetching effect of the proposed cache works well. However, as the SPEC benchmarks show, even a modest sized cache is sufficient to cover much of the executing code. Data caches on the other hand need to cope with much more complex access patterns in order to be effective - often there is no substitute for capacity.

The proposed architecture has two column buffers each 512 bytes long attached to each of the DRAM arrays. These column buffers are dedicated to serving data accesses from the cpu, effectively making a 16KB data cache organized into 512 byte lines and having 2 way associativity. This configuration was simulated in much the same way as the I-cache in order to compare its effectiveness with direct mapped and two-way associative first level caches with a more conventional 32 byte line size. Figure 8 presents the miss rates resulting from these simulations. Each vertical bar presents both the load and the store cache miss probabilities and the combined height is the total cache miss fraction. The bar to the left for each application is the miss rate for the proposed D-cache structure, and the remaining bars represent the conventional caches. The right-most bar for each application illustrates the miss rates after the addition of a small victim cache - we return to this in section 4.4.

Figure 8 shows that the application suite has a significantly more varied D-cache behavior than for an I-cache, which given the reduced temporal and spatial locality of data references compared to instructions, may be expected.

In turn there is a more pronounced difference between the performance of the proposed D-cache structure and conventional cache designs for most of the benchmarks. Those applications which have a high degree of spatial and temporal locality benefit from the prefetching effect of the long lines, while the long lines can also increase the number of conflict misses.

For example, 107.mgrid and 104.hydro2d exhibit markedly reduced D-cache miss rates. over a factor of ten lower for mgrid on the proposed architecture compared to a conventional direct mapped D-cache of the same capacity, and still a factor of 5 lower than a two way associative 256KB conventional cache configuration.

Unfortunately the reverse is also true of other applications; for example, 103.su2cor, 102.swim and 101.tomcatv for which the 512B line size of the proposed cache increases the number of conflict misses to almost a factor of five over those suffered by a conventional cache of the same size.

2. At the time of writing we were unable to direct the fortran compiler to re-layout the code to avoid this contention, but we are working on this.

.

TABLE 2 : Spec'95 Benchmark Components

Benchmark	Description
099.go	Game / Artificial Intelligence: Plays the game Go against itself.
101.tomcatv	Fluid Dynamics/Mesh Generation: Generation of a 2D boundary-fitted coordinate system around general geometric domains.
102.swim	Weather Prediction: Solves system of Shallow Water equations using finite difference approximations.
103.su2cor	Quantum Physics: Computes masses of elementary particles in Quark-Gluon theory.
104.hydro2d	Astrophysics: Solves hydrodynamical Navier Stokes equations to compute galactic jets.
107.mgrid	Electromagnetism: Computes a 3Dpotential field.
110.applu	Math/Fluid-Dynamics: Solves matrix system with pivoting.
124.m88ksim	Simulator: Simulates the Motorola 88100 processor running Dhrystone and a memory test program.
125.turb3d	Simulation/turbulence: Simulates turbulence in a cubic area.
126.gcc	Compiler: cc1 from gcc-2.5.3. Compiles pre-processed source into optimized SPARC assembly code.
129.compress	Compression: Compress large text files (about 16MB) using adaptive Lempel-Ziv coding.
130.li	Interpreter: Based on xlisp 1.6 running a number of lisp programs
132.ijpeg	Imaging: Performs JPEG image compression using fixed point integer arithmetic.
134.perl	Shell interpreter: Larry Wall's perl 4.0. Performs text and numeric manipulations (anagrams / prime number factoring).
141.apsi	Weather: Calculates statistics on temperature and pollutants in a grid.
145.fpppp	Chemistry: Performs multi-electron derivatives.
146.wave5	Electromagnetics: Solve's Maxwell's equations on a cartesian mesh.
147.vortex	A single user O-O database transaction benchmark.Builds and manipulates three interrelated databases. Size is restricted to 40MB for SPEC95
Synopsys	Chip verification operation: compares two logic circuits and tests them for logical identity. The circuits under test contain just over 130 flip-flops.

Each of the benchmark programs was compiled for the SPARC v.8 architecture using the SunPro v.4.0 compiler suite, according to the SPEC base-line rules, and then executed using a simulator derived from SHADE [13]. Cache hit and miss rates were measured, for both instruction and data caches, both for the proposed architecture and for comparable conventional cache architectures.[1]

4.2 Instruction Cache Performance

Figure 7 depicts the instruction cache miss rates for the proposed architecture compared to the miss rates enjoyed by conventionally dimensioned caches.

The left most column for each application depicts the miss probability for an 8KB column buffer cache with 512B lines as proposed, while the remaining bars to the right depict the miss probability for various sizes of conventional direct mapped caches with 32B lines. The figure effectively compares the I-cache miss rate for the proposed architecture with the first level I-cache miss rate for conventional uniprocessors.

Firstly, it is painfully clear from these results that a number of the SPEC 95 benchmarks (namely 110.applu, 129.compress, 102.swim, 107.mgrid, and 132.ijpeg) run very tight code loops which almost perfectly fit an 8KByte cache. Of the remaining 14 applications with non negligeable miss rates three of these (104.hydro2d, 141.apsi, 146.wave5) typically have between a 0.1% and 0.5% miss rate even for an 8KB cache.

The results in figure 7 show that the proposed I-cache with its 512B lines has a significant performance advantage over conventional 1st level caches with their 32byte lines. For almost all of the applications the proposed cache has a lower miss rate than conventional I-caches of over twice the size. In some cases the performance benefits of the longer I cache line size can be very dramatic; for example, in 145.fpppp the miss rate is a factor of 11.2 lower than the conventional cache of the same size, a factor of 8.2 lower than the conventional cache of twice the size (16KB) - note though that the benchmark entirely fits a 64KB I-cache.

The reduced miss rate of the proposed I-cache results directly from the prefetching effect of the long cache line size, combined with the usually high degree of spatial locality found in instruction streams.

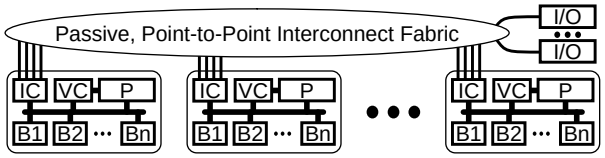
Conventional processor designs are unable to reap the benefits of an increased cache line size because the time it takes to fill such a line introduces severe second order contention effects at the memory interface, however the proposed integrated architecture fills the 512Byte line in a single cycle (after pre-charge and row access) directly from the DRAM array, so these contention effects do not appear. This issue of contention is revisited again in section 3.5

Only two of the Spec benchmarks stand out for their somewhat disappointing I-cache performance; 134.perl has a surprisingly high miss rate - though still lower than the equivalent conventional cache of the same size - because the code is large and has a poor spatial locality. Now, 126.gcc has similar characteristics, but the I-cache miss rates for this application are within 27% those of a 64KB conventional I-cache. Perhaps code profiling to reduce the cache contention may improve the miss rates for perl. The only application to produce a higher miss rate on the proposed architecture was 125.turb3d. This appears to be the result of a direct code conflict between a loop and a function it calls, rather than a general capacity or locality problem. However, while the problem is an artifact of the reduced number of cache lines, it can be removed by a code profiler noting the subroutine being called by the loop[2] - the respective loop and function code can then be re-laid by the compiler or linker to avoid the conflict.

1. A large amount of data was collected from these simulations - not all of which could be presented in this paper. The complete set of results is available as a technical report [9]

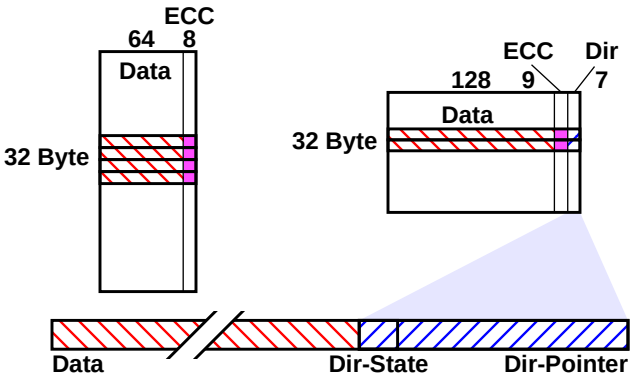
architecture, although the ISA is orthogonal to the concept of processor integration. It is, however, important to point out that an ordinary, general purpose, commodity ISA is assumed. While customization could increase performance, economic considerations strongly argue against developing a new ISA. The R4300i currently consumes 1.5W, which will scale down with the smaller feature size and reduced supply voltage. Therefore it is reasonable to assume that the higher clock frequency will not cause a physically hot chip.

FIGURE 4 : System Overview



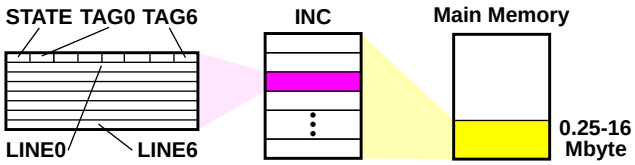
All off-chip communication is handled via the S-Connect system [4], that can operate at 2.5 Gbit/sec in a 0.25 mm process. Four links provide a peak I/O bandwidth of 1.6 Gbytes/sec, which matches the internal memory bandwidth. Notably, all other I/O traffic is handled via the same interconnect medium, that links the memory of all PEs into a common pool of cache coherent shared memory. This means that I/O devices can look like memory and are able to access all memory just like the processor. Due to tight integration between the processor, RMH, RAS and TIC, and because of the smaller faster process, remote memory latencies can be reduced to below 200ns (more conservative figure are used in the performance evalutation).

FIGURE 5 : Directory Structure



As described in [2], cache coherency is mained by mean of a directory structure that is kept in main memory, co-located with the data - avoiding the need for a seperate directory cache. To eliminate the apparent storage overhead for the directory, the directory is encoded in extra ECC bits at the expense of reducing the error correction capability from 1 in 64 to 1 in 128 bits.

FIGURE 6 : Inter-Node Cache Organization



A variable fraction of the memory is reserved for an Inter Node Cache (INC) that holds imported data. This cache is 7-way set-associative (Figure 6) by storing 7 32-byte lines in one 512-Byte Column and storing all the tags in the eighth 32B block. Each INC access requires 1 to 2 extra cycles over a plain (local) main memory access due to the need to check the tags first.

The Victim Cache receives a copy of the most recently accessed 32-byte block of a column buffer whenever a column bufer is reloaded. This data transfer takes place during the time it takes to access the DRAM array on a miss, and thus is completely hidden in the stall time due to the memory access.

4 Uniprocessor Performance

Good multiprocessor scalability on its own is not enough to make a system generally commercially viable. Many of the applications a user may which to execute are not parallelized or even parallelizable. It is important therefore that the integrated processor in the proposed system be capable of executing uniprocessor applications comparably with conventional architectures. Therefore, in this section we concentrate on the performance of the integrated system with uniprocessor applications.

4.1 Methodology : SPEC 95

The current industry accepted standard metric for uniprocessor performance is the SPEC 95 benchmark suite [7]. This suite of programs (described in table 2) is supposed to represent a balanced range of uniprocessor applications, the total execution time of which is used to measure the integer and floating point performance of a processor (and its memory subsystem). As well as using this suite of applications to benchmark the proposed design, the Synopsys application was also added as an example with a large work load of a typical chip design application

Discussions of issues such as processor instruction set architecture, branch prediction or out-of-order execution are essentially orthogonal and beyond the scope of this paper, futhermore these issues involve degrees of complexity not envisioned for the processor under discussion. Instead we concentrate on the novel aspect of the proposal - namely the memory system performance. The simplest first order effect of the proposed design is the cache hit rate afforded.

one layer of metal. However, the upcomming 0.25 µm DRAM process is a very comparable to logic oriented processes.

A significant cost ofproducing DRAMs is the need to test each device, which requires expensive testers. the same holds true for the fabrication of processors. Either device requires complementary support from the tester to function. An integrated processing element greatly reduces the tester requirements, because it is a complete system. All that is required is to download a selftest program. For the system described below, this requires just two signal connections in addition to the power supply.

3 The Design

Given the cost sensitivity of DRAM devices, the design described below tries to optimize the balance between silicon devoted to memory, processor and I/O. The design goal is to add about 10% to the size of the DRAM die, which leading to a processing element with *competitive* performance and a superior costeffectives.

It is possible to devote more area to the processing element in order to achieve superior performance, for example by adding more than one processor, larger caches or a superscaler pipeline. A less cost-effective solution will not enjoy the ecconomy of scale from targeting the mainstream computer applications and fails to leaverage the PC-momentum to serve high end users via scalable, massively parrallel processing with comodity parts. It also reduces practicality; designing competitive DRAMS is as capital intensive as building a high end CPU - an investment that is not taken lightly.

Currently 10% of a 256 Mbit DRAM is about 30 mm^2 which is a bit more than the size of the MIPS R4300i processor when shrunk to a 0.25 µm CMOS process. By using the S-Connect communication fabric [3] for I/O drivers, the number of pads, and interface circuitry is reduced, which adds to the logic area surplus to accomodate about 60K gates for two coherence and communications engines [2].

FIGURE 3 : The Design

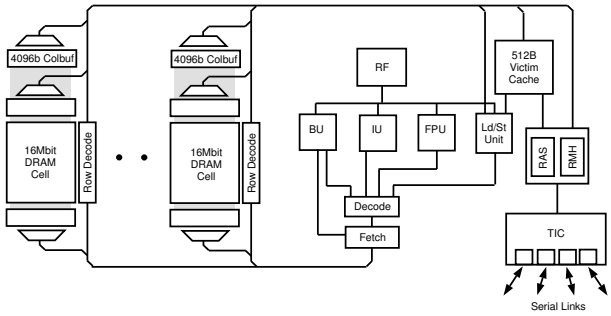


Figure 3 shows a block diagram of the integrated Processor/Memory (P/M) device. The chip is dominated by the DRAM section, which is organized into multiple banks to improve speed (shorter wires have less parasitic capacitance that attenuate and delay the signal from the actual DRAM cell). 16 independent bank controllers are assumed, which is based on existing Rambus [5] and SDRAM [6] configurations. Memory access time is assumed to be 30ns or 6 clock cycles of the 200 Mhz clock. This figure is based on data presented in [7]. Each bank is capable of transfering 4K bits from the sense amplifier array to and from 3 column buffers. These three 512 Byte buffers form the processor instruction and data caches. Two columns per bank are used for a two-way set-associative data cache making a total of 32 512-Byte lines spread across the 16 banks. The remaining 16 column buffers make up a direct mapped instruction cache with 16 512-Byte lines. While normal processors benefit more from larger instruction caches, the fact that there is no line fill-penalty, combined with the much shorter DRAM access latency, cause this 8Kbyte instruction cache to be very effective.

The performance of the 16Kbyte data cache is enhanced with a fully-associative victim-cache [11] of 16 32-byte lines with LRU replacement policy. The victim cache also doubles as a staging area for data that is imported from other nodes.

The nature of large DRAMs require ECC protection to guard against transient bit failures, which encures a 12% size increase if ECC is computed on 64 bit words - the current industry standard. Since all reasonable systems require this level of protection, this 12% overhead should not be counted against this design. Given the cost of the ECC circuitry, this function is performed at the data fetch unit and the load/store unit and not in each bank. Integration has the advantage that ECC checking can proceed in parallel with the processor pipeline, while conventional cpu architectures require that the check has to complete before the data is presented to the processor.

Two independent 64+8 (for ECC) bit data paths connect the column buffers with the processor core, one each for data and instruction access. These busses operate synchrously with the 200 Mhz processor clock and provide 1.6 Gbytes/sec memory access bandwidth each.

All I/O transfer and communication with other processing elements are handled by two specilaized protocol engines, the Remote Memory Handler (RMH) that imports memory from other nodes and the Remote Access Server that takes care of exporting memory. Both can access memory via the data path. The RMH and RAS have been implemented and are described in [2]. The details of their operation is beyond the scope of this paper, but their actual operation is modeled and forms the basis for the multiprocessor evaluation section below. Both CC-NUMA and S-COMA operations are currently supported, and a **COMA-F capability is being added..**

The processor core uses a standard 5 stage pipeline similar to the R4300i or MicroSparc-II. The evaluation presented in this paper was based on the Sparc instruction set

1.1 Background

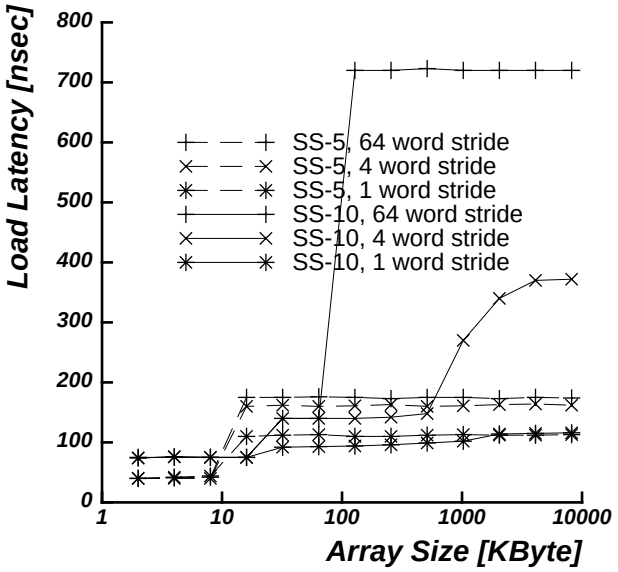
The relatively good performance of Sun's Sparc-Station 5 workstation, with respect to contemporary high-end models, provides evidence for the benefits of tighter memory-processor integration. In the SS-5, the memory controller was actually integrated into the CPU, so that the DRAM devices are directly driven by logic on the processor chip. A separate I/O-bus connects the CPU with peripheral devices, which can access memory only through the CPU chip. Targeted at the low-end of the architecture spectrum, single level, small on-chip caches (16KB L1-I$, 8KB L1-D$) support a single-scalar Micro-Sparc CPU. Compared to the super-scalar SuperSparc CPU, with 2 level caches (16KB L1-D$, 20KB L1-I$, 1MB L2$), of a SS-10/61 the SS-5 has an inferior Spec'92-rating, yet it out-performs the SS-10/61 on a logic synthesis workload (Synopsys) that has a working set of over 50 Mbytes.

TABLE 1 : SS-5 vs. SS-10 Synopsis Performance

Model	Spec'92 Int	Spec'92 Fp	Synt. Run Time
SS-5	64	54.6	32min
SS-10/61	89	103	44min

The reason for this discrepancy is the lower main memory latency of the SS-5, which can compensate for the "slower" CPU. Figure 2 exposes the memory access times for the levels of the cache hierarchy. Codes that miss the large level-2 cache frequently will see lower access time on the SS-5.

FIGURE 2 : SS-5 vs. SS-10 Latencies[1]



The Memory Wall is perhaps the first of a number of impending hurdles which, in the not too distant future, will impinge upon the rapid growth in uniprocessor performance. The pressure to seek further performance though multiprocessor and other forms of parallelism will increase, but these solutions must also over come the Memory Wall.

Forthcoming large scale and ultra-large scale integration technology can address these problems for the first time by finally making feasible the fabrication of a large memory, processor, shared memory controller and interconnection controller together on the same device. This paper presents and evaluates a proposal for such a device.

1.2 The remainder of this paper

The remainder of this paper is organized as follows: Section 2 briefly overviews the technology and trends particularly regarding the sub 0.35μm integration range. This technology will enable the implementation of an integrated microprocessor and memory system. Such a design is presented in section 3. Crucial to the success of any processor design is its uniprocessor application performance this is examined in section 4, and the multiprocessor system performance in section 5. In section 6 we review related work, and conclude in section 7.

2 Technology Characteristics and Trends

The main objection to processor-memory integration is the fact that memory costs tend to dominate and hence economy of scale mandates the use of comodity parts, optimized to yield the most Mbytes/wafer. Attempts to add more capabilities to DRAMs, such as video-buffers (VDRAM), integrated caches (CDRAM), graphics support (FDRAM) and smart, higher performance interfaces (Ram-Bus, SDRAM) were hurt by the extra cost for the non-memory areas. However with the advent of 256 Mbit and 1 Gbit devices [16], memory chips become so large that many computers will have only *one* memory chip. This puts the memory device on equal footing with CPUs and allows them to be viewed as one unit.

In the past, the 7% die-size increase for CDRAMs have resulted in an approximatly 10% increase in chip cost. Ingnoring the many non-technical factors that influnce cost, a 256 Mbit DRAM chips could cost $800 given today's DRAM prices of ~$25/Mbyte. If an extra 10% were added for a processor and extrapolating from the CDRAM case, a processor/memory building block could cost $1000, or $200 for the extra processor. In order to be competitive, such a device needs to exceed the performance of CPU and processor support chips costing a total of $200. It will be shown that such a device would in fact outperform a much more expensive system, in addition to being much smaller, demanding much less power and being much simpler to design in.

Older DRAM technologies were also not suitable for implementing efficient processors. For example, it was not until the 16Mbit generation that DRAMs used more than

1. Memory is accessed in a tight loop and the access time *includes* the loop overhead. The SS-10 has a prefetch unit that hides the memory access time in the case of the small, linear stride.

Missing the Memory Wall: The Case for Processor/Memory Integration

Ashley Saulsbury†, Fong Pong, Andreas Nowatzyk

Sun Microsystems Computer Corporation
†Swedish Institute of Computer Science
e-mail contact: agn@acm.org

Abstract

Current high performance computer systems use complex, large superscalar CPUs that interface to the main memory through a hierarchy of caches and interconnect systems. These CPU-centric designs invest a lot of power and chip area to bridge the widening gap between CPU and main memory speeds. Yet, many large applications do not operate well on these systems and are limited by the memory subsystem performance.

This paper argues for an integrated system approach that uses less power-full CPUs that are tightly integrated with advanced memory technologies to build competitive systems with greatly reduced cost and complexity. Based on a design study using the next generation 0.25µm, 256Mbit DRAM process and on the analysis of existing machines, it is shownthat processor memory integration can be used to build competitive, scalable and cost-effective MP systems.

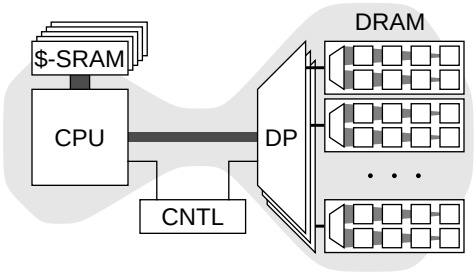
The results from execution driven uni- and multi-processor simulations are presented that show that the benefits of lower latency and higher bandwidth outweight the restrictions on the size and complexity of the integrated processor. In this system, small direct mapped instruction caches with long lines are very effective, as are column buffer data caches augumented with a victim cache.

1 Introduction

Traditionally, the development of processor and memory devices has proceeded independently. Advances in process technology, circuit design, and IC architecture have led to a near exponential increase in processor speed and memory capacity. However, memory latencies have not improved as dramatically and access times are increasingly becoming the limiter of system performance, a phenomenon known as *the Memory Wall* [1]. This problem is commonly addressed by adding several levels of cache to the memory system so that small, high speed, SRAM devices feed a superscalar microprocessor at low latencies. Combined with latency hiding techniques such as prefetching and proper code scheduling it is possible to run a high performance processor at reasonable efficiencies, for applications with enough locality for the caches.

The approach outlined above is used in high-end systems of all the mainstream microprocessor architectures. While achieving impressive performance on applications that fit nicely into their caches, such as the Spec '92 [12] benchmarks, these platforms have become increasingly application sensitive. Large applications such as CAD programs, data bases or imaging applications often fail to meet CPU-speed based expectations by a wide margin.

FIGURE 1 : Compute System Components



The CPU-centric design philosophy has led to very complex superscalar processors with deep pipelines. Much of this complexity, for example out-of-order execution and register scoreboarding, is devoted to hiding memory system latency. Moreover. high-end microprocessors demand a large amount of support logic in terms of caches, controllers and data paths. Not including I/O, a state of the art 10M transistor CPU chip may need a dozen large, hot and expensive support chips for cache memory, cache controller, data path, and memory controller to talk to main memory. This adds considerable cost, power dissipation, and design complexity. To fully utilize this heavy weight processor, a large memory system is required. The effect of this design is to create a bottleneck, increasing the distance between the CPU and memory (Figure 1). It adds interfaces and chip boundaries, which reduce the available memory bandwidth due to packaging and connection constraints; only a small fraction of the internal bandwidth of a DRAM device is accessible externally.

It will be shown that integrating the processor with the memory device avoids most of the problems of the CPU-centric design approach and offers a number of advantages that effectively compensate for the technological limitations of a single chip design.