Architecture and Technology Tradeoffs in the Design of Next-Generation Multiprocessor Servers

David H. Albonesi and Israel Koren
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

Abstract

The design of high performance computing systems requires many design decisions based on performance, cost, power consumption, and possibly other criteria. Decisions made in the early, high-level specification phase are critical to developing a successful product. We describe a methodology which allows the architect to explore alternatives at all design levels for different technology options. We use our approach to explore tradeoffs in the design of high performance multiprocessor servers using next-generation VLSI technology. The performance of a wide range of machine configurations varying in architectural options and technology parameters is explored for several SPEC benchmarks.

1 Introduction

Architectural innovations and improvements in VLSI technology have produced a rapid increase in microprocessor performance. Recently announced microprocessors[6, 9, 14] are projected to exceed 300 SPECint92 and 500 SPECfp92 in uniprocessor performance. The high clock rate and memory bandwidth requirements of these powerful engines, coupled with the lag in a corresponding improvement in board-level packaging and memory technology, creates difficult problems for designers of high performance, multiprocessor servers built around these microprocessors. Designers must squeeze even more performance out of these technologies, while keeping cost within reason. Making optimal architectural and technology decisions in the early, high-level specification phase of machine development is crucial to ensuring that a robust, cost-effective product is developed.

As the microprocessor is only one element of an overall system architecture, it behooves the architect to consider the structure of the overall machine architecture when making decisions on the microprocessor's internal architecture. Knowledge of the available VLSI, packaging, SRAM, and DRAM technologies is necessary to assess how to partition the machine and to assess the impact on performance of selecting various architectural and technology options. A *comprehensive* analysis environment, in which alternatives at all architectural levels in addition to technology options can be explored, is necessary for ensuring that optimal design decisions are made in developing next-generation multiprocessor servers.

To address the complex tradeoffs involved in developing high performance systems, we have developed the **S**ystem **T**radeoff **A**nalysis **T**ool**s**et (STATS), which integrates trace-driven simulation, analytical modeling, and Spice simulation tools that are used to analyze all architectural levels in the context of the underlying VLSI and packaging technologies. The semi-automatic nature of STATS facilitates the rapid exploration of a wide variety of architectural options, while the integration of all architectural levels and the incorporation of technology limitations ensures accuracy. In this paper, we use STATS to evaluate the complex, multidimensional design space of a multiprocessor server, and thereby demonstrate how the tool can be used to pare down the design space to a more managable realm.

2 STATS Overview

Figure 1 shows the overall structure of STATS. STATS contains over 600 architectural and technology parameters that can be varied by the architect. Architectural parameters span the processor, cache hierarchy, multiprocessor interconnect, and main memory system. Technology parameters include ASIC and microprocessor VLSI technology, packaging at all levels (chip, daughtercard, board, backplane), and the SRAM and DRAM technology. In addition, the system topology describes the overall connection of the various system components. For example, the processor, memory, and I/O modules may all lie on the same board, be packaged in a daughtercard-motherboard arrangement, or in a board-backplane configuration. The architect can modify numerous parameters to ex-

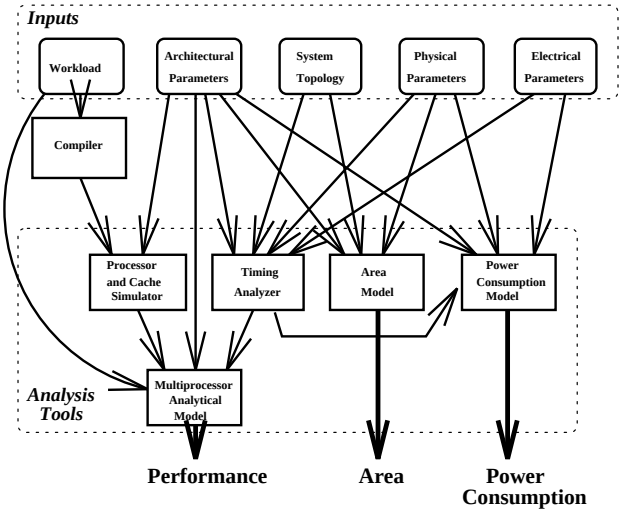

Figure 1: STATS Tool Flow

periment with cost/performance tradeoffs with different types of packaging.

The Workload section of STATS consists of benchmarks that are compiled for the Processor and Cache Simulator and parameters for the Multiprocessor Analytical Model. An example of the latter is the probability of a cache store miss being found dirty in another cache. The compiler is a modified version of the Multiflow Trace Scheduling compiler[8] that is targetted to the Digital Alpha architecture.

At the heart of STATS are five main analysis tools. The Processor and Cache Simulator is a trace-driven simulation model of a pipelined superscalar CPU and cache hierarchy. The simulator uses a modified Alpha 21064 model that is highly parameterized so that a broad range of architectures can be explored. The Timing Analyzer determines the cycle time of the processor and off-chip logic (bus and main memory). A expanded version of Wilton's cache cycle time model[16], which includes pipelining and multi-ported arrays, is used to determine the cycle time of the cache hierarchy. The cache access operation is broken down into a set of "substages", such as address decode, array access, tag compare, data output, etc. Each substage can become an individual stage in the cycle time model, or be combined with adjacent substages to form a stage. The model can have 1 to 4 data stages, and 1 to 5 (direct mapped) or 1 to 6 (set associative) tag stages for on-chip caches. For cache arrays with less than the maximum number of stages, the model calculates substage delays and recursively finds the combination of substages that minimizes cycle time.

For off-chip tag or data arrays, Spice modeling is performed to determine the delays of the address and data paths. Spice runs of the multiprocessor bus configuration are then performed and analyzed. The off-chip clock multiple that the external logic must operate at is then determined by dividing the bus latency (including clock to Q register delay, setup time, and clock skew) by the processor cycle time, and rounding up to the next highest integer[1]. A higher off-chip clock multiple causes more processor cycles to be needed for some operations, for example to traverse the system bus. The Timing Analyzer then uses timing information specified in the Architectural Parameters to determine the number of processor cycles needed to perform various bus operations. Overheads for main memory operations are similarly determined using Spice runs of control and datapath delays.

The Multiprocessor Analytical Model is a detailed Mean-Value-Analysis (MVA) multiprocessor model that extends previous MVA models in several ways[1]. Modern superscalar processors using latency reduction techniques are modeled, and fine details about the architecture (for example, state machine overheads for each operation type at each architectural level) can be specified. Analytical modeling was chosen over trace-driven simulation since MVA models have been shown[4] to provide good correlation with simulation for throughput-oriented (server) environments without incurring the long runtimes of simulation. Using analytical modeling allows many different uniprocessor and multiprocessor configurations to be explored quickly and with sufficient accuracy. Once the design space has been pared down, simulation can be used on the smaller number of candidate configurations.

The Area and Power Consumption Models provide estimates based on the system organization and electrical and physical parameters. The clock rate information from the Timing Analyzer also factors into the power consumption calculations.

In order to produce results quickly, the system builds up a database for the time-intensive portions of the analysis, namely the Processor and Cache Simulator, the Spice simulators in the Timing Analyzer, and the Workload/Compiler combination. A database listing and result files for previously performed simulation and compilation runs are maintained, and a lookup is done before new runs are made. Thus, although initially runtime can be long (a half an hour or more), once a database of past runs is built up, runtime can be as short as 15 seconds on a 175MHz Alpha 3000 Model 600 workstation.

[1]Non-integer multiples, e.g. 1.5, are used in some machines[3], but we assume in this paper that integer multiples are required.

It should be made clear that once the architect sets the input parameters, STATS performs the entire analysis without the need for manual intervention, providing both final results as well as intermediate information (such as the bus cycle delays produced by the Timing Analyzer). Thus, the designer is freed from the arduous and time-consuming task of setting up inputs for and running individual tools. Since extensive performance information is provided by STATS, the architect can also set up scripts that perform multiple runs to search the design space automatically.

The Compiler and Processor Simulator are currently being integrated into STATS. In the interim, we are using traces of the SPEC benchmarks to drive the models to explore cache and multiprocessor tradeoffs not possible with other tools. The results reported in the next section were obtained using this intermediate version of STATS.

3 Tradeoffs in Designing High Performance Multiprocessor Servers

The multidimensional design space of next-generation multiprocessor servers presents a difficult challenge for the designer. The sheer size of the design space and the interplay between different architectural levels and between architecture and technology makes it extremely difficult for the designer to make optimal design choices. Due to its comprehensive nature and fast execution speed, STATS allows the architect to vary a wide range of parameters and quickly gauge the performance of each configuration. In the next section, we describe some of the tradeoffs involved in the design of next-generation servers, and describe the range of parameters that we consider.

3.1 Design Space and Assumptions

Most current multiprocessor server systems[3, 5] use a system bus routed on a central backplane on which processor, memory, and I/O boards can be added. This arrangement meets upgradability requirements and the aggressive main memory and I/O objectives of large configurations. To continue to meet these criteria, we expect that next-generation multiprocessors will use a similar system organization. Like the AlphaServer 2100[5], we assume from 1 to 4 processors and that two I/O modules reside on the system bus. We investigate using a maximum of 2 or 4 memory modules in the system. The former reduces bus length and loading, while the latter reduces memory contention through greater interleaving.

As current leading-edge microprocessors are implemented in 0.5 micron technology, it is expected that next-generation microprocessors will use a minimum feature size of 0.35 microns. This high level of integration will permit the implementation of twice as much on-chip cache as today's processors. As the Alpha 21164[14] has the largest on-chip L2 cache (96kB) of current microprocessors implemented in 0.5 micron technology, a 0.35 micron chip should easily be able to support 192kB of on-chip L2 cache (with small on-chip L1 caches). Alternatively, larger off-chip L2 caches can be designed using fast SRAMs (32k×8[10] and 128k×8[11] parts from Motorola, assumed to have access times of 4ns and 6ns, respectively), and dual address buses to minimize off-chip delays. The performance tradeoff between these options is faster on-chip access time versus lower cache miss rates. As it is unclear which organization provides best uniprocessor and multiprocessor performance, especially considering a variety of other parameters, we analyze both on and off-chip cache options[2]. In all cases, however, we assume that the L2 cache tags are integrated on-chip. This permits faster tag lookup, and dual porting of the tags for bus snooping. We also vary the degree of on-chip array pipelining, and the width of the L2 cache data bus. A higher degree of L2 cache pipelining potentially reduces cycle time but at the expense of increased latency. A wider cache data bus reduces latency but possibly at the expense of increased cycle time. We assume that the datapath between the L2 controller and the bus controller matches the width of the L2 data bus.

Designing a system bus with adequate throughput is key to ensuring good multiprocessor scalability. Designing high performance buses becomes a greater challenge as microprocessor clock rates continue to increase. Although ASIC circuit speeds increase as well with new microprocessor generations, printed circuit board trace delays and board packaging technology remain relatively invariant. In addition, more aggressive heat sinks may be needed to dissipate heat from faster microprocessors, increasing spacing between adjacent boards. Thus, for a given bus topology, the off-chip clock multiple that the bus logic runs at will invariably increase as microprocessor frequencies continue to rise. Current generation servers use careful electrical design, wide buses, and in some cases, dual-processor boards to reduce bus loading and increase throughput. This is usually done after the microprocessor has been designed and fabricated. As microprocessor clock frequencies continue to rapidly increase and multiprocessing becomes more prevalent, additional measures may need to be taken in the microprocessor design

[2] An on-chip L2 cache coupled with an off-chip L3 cache[14] is also an option, but we don't consider this in this paper.

Parameter	On-Chip Values	Off-Chip Values
Size	128kB, 192kB	256kB, 512kB, 1MB, 2MB
Associativity	2, 3-way	1-way
SRAMs used	—	32kB, 32kB, 128kB, 128kB
Stages	3-5	4-5 (tag only)
Data Width	128-256	64, 128, 64, 128
Tag Ports[4]	1 or 2	1 or 2

Table 1: L2 Cache Parameters and Range of Values

phase to ensure adequate multiprocessor bus throughput. These may include:

- Integrating the bus controller onto the microprocessor[3]. This eliminates the bus control ASIC and the associated inter-chip delays. In addition, if the L2 cache tags are on-chip, they may be dual-ported to eliminate duplicate snooping tags.
- Integrating the memory controller (MC) onto the microprocessor. This reduces the length and capacitive loading of the bus, but at the possible expense of less flexible upgrading capability.

These options, as well as using different width buses and single or dual processor boards, are examined. As mentioned above, incorporating two processors on a board sharing a common bus interface ASIC may reduce bus loading. This comes at the expense of greater latency and more coherency overhead, as bus operations require microprocessor to bus controller interchip delays, and a duplicate set of snooping tags must be maintained. However, the duplicate set of tags permits the main L2 cache tags to be single-ported which may reduce cycle time.

In designing main memory systems, different speed grade DRAMs present different cost/performance tradeoffs to the designer. We examine the performance difference of using 16Mb DRAMs with 40ns and 50ns access times, and 90ns and 110ns cycle times, respectively. These speeds are 10ns faster than those found in [12, 13]. We also examine the difference in performance of two different memory data bus widths (144 bits and 288 bits, using 8 bits of ECC for each 64 bits of data). We use 4M×4[12] and 2M×8[13] DRAMs, respectively, for these two bus widths.

Tables 1 and 2 summarize the server design space explored. The broad range of parameters investigated results in 960 different configurations to analyze. A complete listing of architectural and technology parameters is being prepared[2].

[3]This is already done on some microprocessors, e.g., [9].

[4]The value depends on whether a single or dual processor board is used.

Element	Parameter	Values
Bus	Width	128, 256
Bus	Processors/Board	1, 2
Bus	Memory Controller	separate, integrated
Bus	Max Memory Nodes	2, 4
Memory	Width	128, 256
Memory	DRAM access times	50ns, 40ns
Memory	DRAM cycle times	110ns, 90ns

Table 2: System Bus and Main Memory Parameters and Range of Values

3.2 Workload and Performance Measures

For multiprocessor servers, two popular benchmarks are SPECrate_int92 and SPECrate_fp92. For SPECrate_int92, each SPECint92 benchmark is run with concurrent copies of itself. SPECrate_fp92 is run in a similar manner. Thus, these benchmarks estimate the coarse-grained multiuser workloads that predominate on server platforms. STATS emulates the SPECrate benchmarks as follows. Cache hierarchy trace-driven simulations of each of the SPECint92 and SPECfp92 benchmarks are run and performance statistics gathered. The multiprocessor analytical model is then run for each benchmark using these statistics and designer-supplied values for cache coherence parameters (e.g., the probability that upon a load miss, the block is found modified in another cache). As these parameters are typically well-known, having been measured on previous generation machines[3], the method used by STATS provides sufficiently accurate results in orders of magnitude less time than multiprocessor simulation.

For space reasons, in this paper we examine three of the SPECfp92 benchmarks: nasa7, su2cor, and swm256. Our traces range in size from 87 million to 99 million references. The cache miss rate trends that we observe using these benchmarks are in agreement with previous studies such as [7]. Due to space constraints, we only present results for one and four processor systems, although runs were made for all configurations.

With three benchmarks and the range of parameters described previously, we have 2880 different analyses to perform. As a cache simulation database was already built up from a previous analysis, STATS was able to perform this task in less 20 hours on a 175MHz Alpha 3000 Model 600 workstation.

3.3 Performance Results

We now evaluate the performance impact of the design alternatives for each of the three benchmarks. Our goals are to pare down the design space to a more managable number of configurations, and to understand the impact of various alternatives on per-

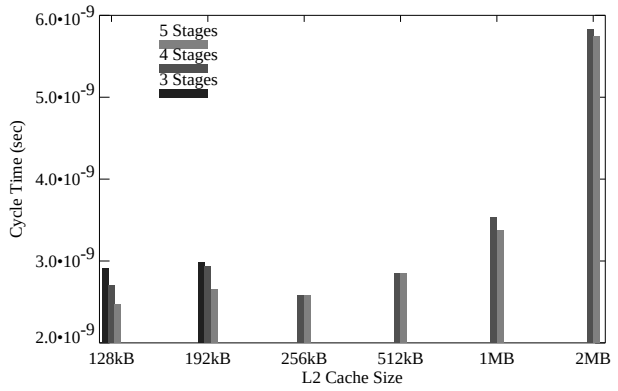

Figure 2: L2 Cache Cycle Time Variation with Pipelining Degree

formance. We list the configurations with the best uniprocessor and multiprocessor performance for each benchmark, and examine the effects of L2 cache, system bus, and main memory features in turn. We also discuss the best configurations in terms of performance averaged over the three benchmarks.

Tables 3 and 4 list the top five performing uniprocessor and multiprocessor system configurations for nasa7 and swm256. The results for su2cor are similar to those for nasa7 except that 3 L2 pipeline stages are used instead of 5. Configurations are rated in instructions per second (denoted IPS in the tables), and are ranked by uniprocessor performance as the first criteria and multiprocessor performance as the second. A "U" in the last column indicates that the configuration is among the top configurations in uniprocessor performance, an "M" represents a configuration which excels in multiprocessor performance, and "U,M" means that the configuration is among the top configurations in both. In addition, efficiency (denoted Eff), calculated by dividing the 4-processor IPS by 4 times the uniprocessor IPS, is shown to gauge the scalability of each configuration under the various workloads.

Evaluating L2 cache options for each of the benchmarks, for both the nasa7 and su2cor benchmarks, the 192kB configurations provide the best uniprocessor and multiprocessor performance. As mentioned above the optimum number of L2 pipeline stages for nasa7 is 5 and 3 for su2cor. Figure 2 shows how varying degrees of pipelining affects cycle time for the L2 caches studied. Looking at the 192kB cache organization, we see that pipelining has a significant effect on cycle time. However, su2cor differs from nasa7 in that it exhibits much larger L1 cache miss rates; thus, the performance improvements gained from the lower latency of the 3-stage L2 cache organizations benefits su2cor more than the reduced cycle time of the 4 and 5-stage configurations. A similar argument holds for the L2 cache data width. Although the 132-bit wide organizations are more prevalent, su2cor does benefit in some cases from the wider 264-bit caches coupled with a 4-stage pipeline which tends to offset some of the cycle time increase incurred. For both nasa7 and su2cor, the same L2 cache configurations in general provide the best uniprocessor and multiprocessor performance.

The best performing L2 cache configurations for the swm256 benchmark (Table 4) contrast markedly with those for nasa7 and su2cor. As noted in [7], the miss rates for swm256 stay relatively invariant until the cache size gets very large. From our simulations, the L2 cache miss rate does not tail off until the 2MB cache size is reached. As we see in Figure 2, the cycle time of the 2MB configuration is much larger than the other configurations, which offsets the miss rate benefit. The reason for this increase in cycle time is the Decode2 substage delay of the on-chip tag array which increases rapidly as the cache size reaches 256kB and beyond. Thus, for uniprocessor IPS, since the miss rates of the other configurations are equally poor, the best cycle time configurations in general deliver the best performance results for swm256. Unlike nasa7 and su2cor, the configurations that produce the best multiprocessor performance have in general larger L2 caches than the best uniprocessor configurations for swm256, which exhibit poor efficiency due to very high bus waiting times caused by bus saturation. The 512kB configurations, which have slightly lower L2 cache miss rates than the smaller on-chip caches (and a wider cache datapath than the 256kB organizations) provide the best balance between cycle time and system throughput for multiprocessor performance. The 2MB configurations (not shown in the table) provide greatly improved efficiency (96.4%) due to their much lower L2 cache miss rates, resulting in better system throughput when running swm256. The difference in required L2 cache size for uniprocessor and multiprocessor configurations for swm256 justify the differences in L2 cache sizes found in single processor workstations and multiprocessor servers. For example, the AlphaStation 400 4/233 single processor workstation contains 512kB of L2 cache while the AlphaServer 2100[5] has up to 4MB of L2 cache per processor. As we have shown, larger caches are needed in a multiprocessor server to ensure that bus waiting times are kept to a minimum.

Table 5 presents averaged results for the three

L2 Cache			System Bus				Memory System		Performance			U or M
Size	Stages	Data Width	Width	CPUs/ Board	Max Mem	Integr MC	Data Width	DRAM Access	IPS ($\cdot 10^6$) 1 CPU	IPS ($\cdot 10^6$) 4 CPUs	Eff	
192kB	5	132	256	1	4	yes	256	40.00	402.818	1595.966	0.991	U,M
192kB	5	132	128	1	4	yes	256	40.00	402.701	1588.875	0.986	U
192kB	5	132	256	1	4	yes	128	40.00	402.460	1594.576	0.991	U,M
192kB	5	132	256	1	4	yes	256	50.00	402.407	1593.480	0.990	U,M
192kB	5	132	128	1	4	yes	128	40.00	402.343	1587.508	0.986	U
192kB	5	132	256	1	4	yes	128	50.00	402.049	1592.095	0.990	M
192kB	5	132	256	2	4	yes	256	40.00	401.974	1591.285	0.990	M

Table 3: Best Performing Configurations for nasa7

L2 Cache			System Bus				Memory System		Performance			U or M
Size	Stages	Data Width	Width	CPUs/ Board	Max Mem	Integr MC	Data Width	DRAM Access	IPS ($\cdot 10^6$) 1 CPU	IPS ($\cdot 10^6$) 4 CPUs	Eff	
128kB	5	132	256	1	4	yes	256	40.00	271.393	577.711	0.532	U
128kB	5	132	256	1	4	yes	128	40.00	265.400	576.325	0.543	U
192kB	4	264	256	1	4	yes	256	40.00	264.539	596.161	0.563	U
128kB	5	132	256	1	4	yes	256	50.00	263.137	572.696	0.544	U
192kB	3	132	256	1	4	yes	256	40.00	262.918	593.441	0.564	U
512kB	4	144	256	2	4	yes	256	40.00	229.382	645.298	0.703	M
512kB	4	144	256	2	4	yes	128	40.00	227.139	642.539	0.707	M
192kB	5	132	256	2	4	yes	256	40.00	241.093	641.692	0.665	M
512kB	5	144	256	2	4	yes	256	40.00	223.783	641.523	0.717	M
192kB	5	132	256	2	4	yes	128	40.00	236.984	639.128	0.674	M

Table 4: Best Performing Configurations for swm256

L2 Cache			System Bus				Memory System		Performance			U or M
Size	Stages	Data Width	Width	CPUs/ Board	Max Mem	Integr MC	Data Width	DRAM Access	IPS ($\cdot 10^6$) 1 CPU	IPS ($\cdot 10^6$) 4 CPUs	Eff	
192kB	3	132	256	1	4	yes	256	40.00	314.884	1067.663	0.848	U,M
128kB	5	132	256	1	4	yes	256	40.00	314.814	993.748	0.789	U
192kB	3	132	256	1	4	yes	128	40.00	313.071	1065.275	0.851	U,M
192kB	3	132	128	1	4	yes	256	40.00	312.935	1001.792	0.800	U
192kB	3	132	256	1	4	yes	256	50.00	312.111	1056.742	0.846	U
192kB	5	132	256	2	4	yes	256	40.00	300.610	1069.204	0.890	M
192kB	5	132	256	2	4	yes	128	40.00	298.753	1066.723	0.893	M
192kB	5	132	256	2	4	yes	256	50.00	297.772	1057.148	0.888	M

Table 5: Best Performing Configurations for Averaged Results

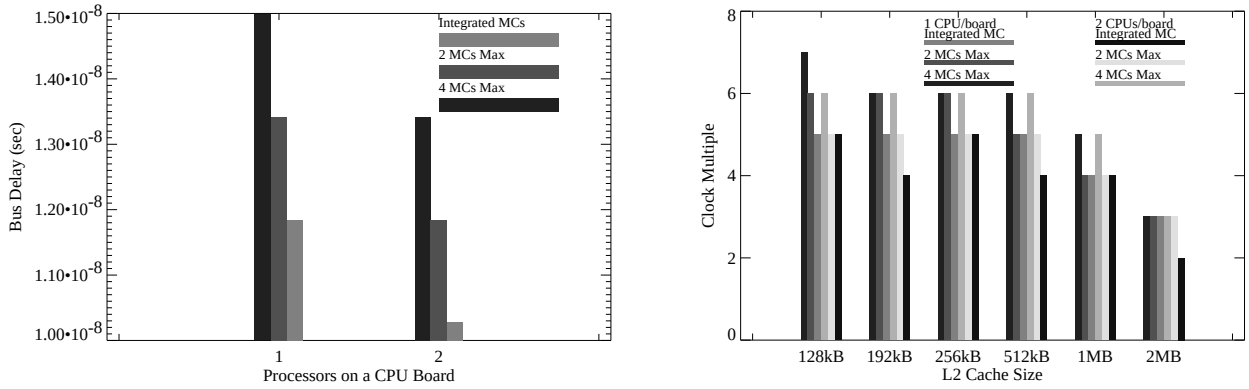

Figure 3: (a) Bus Delay and (b) Clock Multiple for Various Bus Options

benchmarks. Due to the dominance of the 192kB configurations for the nasa7 and su2cor benchmarks, these provide the best overall results. The modest efficiency of these configurations is due to the swm256 benchmark. Although our results indicate that the 192kB configurations provide the best overall balance between cycle time and system throughput, this configuration is not the best choice for the swm256 benchmark running on a multiprocessor configuration. Implementing a 2MB, off-chip L3 cache to back up the 192kB on-chip cache would appear to improve the performance of swm256 by significantly reducing bus traffic.

Turning to system bus integration alternatives, many of these boost the performance of all three benchmarks considerably, especially swm256, whose L2 cache miss rates are significantly higher than nasa7 and su2cor. For swm256, using a 256 bit wide system bus increases the 4-CPU IPS by roughly 21%, and greatly improves efficiency. The integrated memory controller (MC) option is even more beneficial, increasing multiprocessor performance in excess of 30%. Figure 3, which shows bus delay and system clock multiple for the various bus options, shows why this is the case. Each integration option has a significant impact on bus delay. Integrating the MC reduces bus delay of the single-processor board, 4-memory-module configuration by roughly 21%. It should be noted however, that a reduction in bus delay does not necessarily translate into a faster bus frequency. This depends on whether the reduction in bus delay allows the off-chip interface to operate at a lower processor clock multiple for the particular configuration.

While nasa7 and su2cor do not benefit from dual processor boards, this bus integration measure greatly improves multiprocessor IPS for the swm256 benchmark. The performance of swm256 is highly dependent on bus throughput; the reduction in bus waiting time obtained from the faster operating dual processor board bus organization offsets the increased latency imposed by the addition of the bus control ASIC. For nasa7 and su2cor, the additional latency imposed by the separate bus control ASIC offsets the bus delay reduction obtained. Based on the results for swm256, we expect that servers with larger numbers of processors than we have investigated would be more likely to require dual processor boards, as bus throughput would be more crucial to achieving good performance for cache-intensive programs. The Hewlett-Packard T500 server[3], which contains up to 12 processors, uses dual processor boards because of this requirement. For our configurations, the averaged results in Table 5 show that the decision of whether to implement a single processor board with the bus controller integrated onto the microprocessor or a dual processor board with a common bus control ASIC is not clear-cut. While single processor boards provide best uniprocessor performance for all benchmarks and best multiprocessor performance for nasa7 and su2cor, the dual processor boards offer a significant performance improvement over single processor boards for swm256. The best choice depends on the relative importance of these and other criteria such as cost, modularity, and power consumption. The main point is that using the STATS methodology provides the information necessary to make the best decision.

The memory enhancements (a wider data bus and faster DRAMs) provide very little performance benefit for all benchmarks. Our choice of a 32-byte block size, which with a 128-bit memory data bus requires only two memory accesses, partially explains why the performance improvement is so small. With a larger choice of block size, more significant improvements may be obtained.

In summary, for the benchmarks we have run, our results indicate that future high end microprocessors, to be integrated into high performance multiprocessor servers, should continue the cache hierarchy strategy established in the 21164 microprocessor. A pipelined, on-chip 192kB L2 cache should be implemented while providing the capability for a larger optional off-chip L3 cache for cache-intensive benchmarks like swm256. We have also shown that in order to maximize the multiprocessor performance of next-generation servers, attention needs to be paid in the microprocessor design phase to bus throughput enhancing options, like integrating the memory controller onto the microprocessor, and providing the capability to link two processors with a common bus interface ASIC, to reduce bus propagation delay.

4 Conclusions and Future Work

As VLSI integration levels continue to increase, the design of high performance systems becomes increasingly complex. We have developed STATS, a comprehensive environment for making design tradeoffs, to address this issue. Using STATS, we have explored tradeoffs in the design of next-generation multiprocessor servers, for a range of parameters spanning the cache hierarchy, system bus, and main memory system. Results for three of the SPEC benchmarks demonstrate how STATS can be used to analyze complex design spaces.

We are currently integrating a pipeline simulator and compiler into STATS. This will allow performance

and technology tradeoffs to be made at all architectural levels. Area and power consumption models are being developed to include these important criteria in design decisions. Although STATS is effective at quickly analyzing a design space, more intelligent search methods may be necessary as the number of varied parameters increases. An approach like [15], in which a genetic algorithm is used to search the design space, is an area for future investigation.

Acknowledgements

The authors wish to thank Digital Equipment Corporation, especially Tryggve Fossum, Michael Adler, Joel Emer, Geoff Lowney, Bob Nix, and David Webb, for providing the compiler and pipeline simulator for STATS. Steve Wilton provided valuable information on his cache cycle time model, as well as advice on expanding it. We also thank the members of the Tracebase project at New Mexico State University for making the traces used in this paper available.

References

[1] D.H. Albonesi and I. Koren, "An Analytical Model of High Performance Superscalar-Based Multiprocessors," *3rd International Conference on Parallel Architectures and Compilation Techniques (PACT'95)*, pp. 194-203, June 1995.

[2] D.H. Albonesi and I. Koren, "A Methodology for the Comprehensive Exploration of Microprocessor Design Alternatives," Technical Report, Electrical and Computer Engineering, University of Massachusetts, to appear, 1995.

[3] T.B. Alexander, et al, "Corporate Business Servers: An Alternative to Mainframes for Business Computing," *Hewlett-Packard Journal*, Vol. 45, No. 3, pp. 8-30, June 1994.

[4] M. Chiang and G.S. Sohi, "Experience with Mean Value Analysis Models for Evaluating Shared Bus, Throughput-Oriented Multiprocessors," *SIGMETRICS'91*, pp. 90-100, May 1991.

[5] F.M. Hayes, "Design of the AlphaServer Multiprocessor Server Systems," *Digital Technical Journal*, Vol. 6, No. 3, Summer 1994.

[6] Hewlett-Packard Corporation, "HP Announces World's Most Powerful Microprocessor," Press Release, March 6, 1995.

[7] J.D. Gee, et al, "Cache Performance of the SPEC92 Benchmark Suite," *IEEE Micro*, pp. 17-27, August 1993.

[8] P.G. Lowney, et al, "The Multiflow Trace Scheduling Compiler," *Journal of Supercomputing*, No. 7, pp. 51-142, 1993.

[9] MIPS Technologies, Inc, "R10000 Microprocessor Product Overview," October 1994.

[10] Motorola Inc., MCM6706R 32k×8 Bit Static Random Access Memory Product Preview, Fast Static RAM Databook, pp. 2-15 to 2-26, 1993.

[11] Motorola Inc., MCM6726A 128k×8 Bit Static Random Access Memory Product Preview, Fast Static RAM Databook, pp. 2-39 to 2-50, 1993.

[12] NEC Electronics, Inc., uPD4216400 4,194,304 × 4-bit Dynamic CMOS RAM, Memory Products Databook, pp. 8d-1 to 8d-18, 1993.

[13] NEC Electronics, Inc., uPD421x800 2,097,152 × 8-bit Dynamic CMOS RAM, Memory Products Databook, pp. 8h-1 to 8d-24, 1993.

[14] P. Rubinfeld, "An Overview of the 21164 Alpha AXP Microprocessor," *Hot Chips VI*, August 1994.

[15] T.J. Stanley and T. Mudge, "Systematic Objective-driven Computer Technology Optimization," *16th Conference on Advanced Research in VLSI*, March 1995.

[16] S.J.E. Wilton and N.P. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches," Digital WRL Research Report 93/5, July 1994.