Tradeoffs in the Design of Single Chip Multiprocessors

David H. Albonesi and Israel Koren

Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, USA

Abstract: By the end of the decade, as VLSI integration levels continue to increase, building a multiprocessor system on a single chip will become feasible. In this paper, we propose to analyze the tradeoffs involved in designing such a chip, and specifically address whether to allocate available chip area to larger caches or to large numbers of processors. Using the dimensions of the Alpha 21064 microprocessor as a basis, we determine several candidate configurations which vary in cache size and number of processors, and evaluate them in terms of both processing power and cycle time. We then investigate fine tuning the architecture in order to further improve performance, by trading off the number of processors for a larger TLB size. Our results show that for a coarse-grain execution environment, adding processors at the expense of cache size improves performance up to a point. We then show that increasing TLB size at the expense of the number of processors can further improve performance.

Keyword Codes: C.1.2; C.4; C.5.4
Keywords: Multiprocessors; Performance of Systems; VLSI Systems

1 Introduction

VLSI integration levels continue to rapidly increase, so much so that it has been predicted that by the end of the decade, a one inch square chip with 0.25 micron technology will be available[17]. At this level of integration, it will become possible to build a multiprocessor system with several of today's microprocessors on a single chip. Microprocessor designers will have a wide design space to explore, and many tradeoffs to make between processor architecture, cache hierarchy and TLB organization, interconnect strategies, and incorporating multiple processors on the chip. In this paper, we begin to address the design of such *high integration microprocessor architectures*. In particular, we evaluate performance tradeoffs in allocating chip resources to larger caches versus more processors. We also investigate how elements of the processor architecture (in particular TLB size) affect design decisions.

The rest of this paper is organized as follows. First we discuss the multiprocessor organization and establish performance metrics. We then calculate fixed area overheads for I/O logic/pads and the system bus. Next, we determine candidate system organizations

and discuss our modeling approach. We then determine processing power, cycle time, and total system performance for each configuration. Lastly, we present our conclusions and discuss possible future extensions to our work.

2 System Organization and Performance Metrics

The overall architecture that we consider is a shared memory multiprocessor system containing n processors, each of which has a private cache. A system bus is used as the interconnect structure between the caches and the main memory, which may consist of several interleaved modules. With the technology predicted to be available at the end of this decade, designers will be able to place this entire structure (for a limited number of processors and with the exception of the main memory DRAMs), onto a single chip[1].

The execution environment that we assume is a coarse-grain, *throughput-oriented*, parallel environment. The system might be for example a server running Unix with many X terminals connected to it. Each of these X sessions runs separate user tasks and applications, and shares primarily operating system code and data structures and common application code. Thus the amount of shared data is minimal, and the vast majority of the time the processors are accessing private data.

The performance metric that we wish to maximize is *total system performance* which can be expressed as *processing power/cycle time* where *processing power* is defined by $n/(CPI \cdot instr)$. Here n is the number of processors in the system, CPI is the cycles per instruction rating for each processor, and $instr$ is the number of instructions executed in the running of the program. This last parameter is a function of the instruction set architecture. Since this paper does not focus on instruction set architecture design issues, we eliminate this term and thus obtain n/CPI for processing power.

3 Tradeoffs Between Cache Size and Number of Processors and the Effect of TLB Size

Having established the general system organization, execution model, and performance evaluation criteria, we now examine some of the tradeoffs involved in designing single chip multiprocessors. We consider the problem of whether to use the available chip area for enlarging the caches or for adding additional processors. We focus on these two architectural parameters (the size of the caches and the number of processors) since they have such a great impact on performance. Once we have made area tradeoffs for these parameters and have a region of the design space narrowed down, we can then address parameters which have less impact on performance. In this paper, we address the size of the TLB.

The physical aspects of our study are based on the dimensions of the Alpha 21064 microprocessor[5, 13]. We scale the processor core and cache dimensions of the 21064 to 0.25 micron technology, and assume the use of a one inch square die. We then use this data to obtain candidate multiprocessor configurations which vary in cache size and number of processors.

[1] An experimental version of such a chip has already been designed and fabricated in 0.30 micron technology[10].

3.1 Bus and I/O Overhead

We use an aggressive bus design, consisting of separate 64-bit wide address and 128-bit wide data buses distributed to all the caches as well as the main memory controller on the chip. Based on previous implementations of bus-based multiprocessors, an aggressive bus design is necessary for a moderately large (8-16) number of processors. We allocate roughly 20% of the chip area for the bus, external interface, and I/O pads. Based on empirical measurements, approximately 17% of the area of the 21064 is allocated to external control and I/O pads. Due to the large number of I/O and power signals expected on our chip, and because I/O pad size may not scale as well as transistor sizes, we assume the same overhead on our chip. To determine the area consumed by the bus, we scale the dimensions of the second layer of metal (Metal 2) on the Alpha chip, since this is the wider of two metal layers used for general signal distribution. (The third layer is primarily used for power and clock distribution.) Metal 2 has a width of 0.75μm and a pitch of 2.625μm. We consider two means of scaling for comparison purposes: ideal scaling and $\sqrt{S_c}$ scaling where S_c is the scaling factor. The latter has been suggested in [1] in order to reduce propagation delays as technology is scaled. In our case, since we are scaling a 0.75 micron technology to 0.25 microns, S_c has a value of 3. We assume an additional 15 signals beyond those for address and data for arbitration, signaling of operations on the bus, and acknowledgements. Thus our total bus signal count is 207. We obtain for the width of the bus using ideal scaling $(0.75 \cdot 10^{-3} + 2.625 \cdot 10^{-3}) \cdot 207/3 = 0.23$mm and using $\sqrt{S_c}$ scaling $(0.75 \cdot 10^{-3} + 2.625 \cdot 10^{-3}) \cdot 207/\sqrt{3} = 0.40$mm.

Assuming the bus runs almost the entire length (20mm) of the chip, it consumes 0.7% and 1.2% of the chip area with ideal and $\sqrt{S_c}$ scaling, respectively. Scaling the metal dimensions for the bus by $\sqrt{S_c}$ instead of ideally has a negligible impact on the overall chip area. Adding the area for I/O gives us 17.7% and 18.2% for ideal and $\sqrt{S_c}$ scaling, respectively. In order to account for additional area lost due to the routing problems inherent in such a wide bus, we boost our overall figure for the area consumed by the I/O interface and bus to 20%.

3.2 Candidate Configurations

To determine candidate configurations, we proceed as follows. First, we empirically determine the area of the caches (16kB total) and processor core (the chip minus the caches and I/O pads) of the 21064. We scale (using ideal scaling) these dimensions to 0.25 micron technology, and then determine the fraction of a 0.25 micron, one inch square die consumed by the scaled processor and caches. The area of 32kB, 64kB, and larger caches is determined by using multiples of the base 16kB cache area. This is to a first order, consistent with area models described in [14, 15], especially for large caches where the size of the data area dominates the overall size.

We combine the dimensions of these caches with the dimension of the processor core to construct candidate processor/cache organizations. We then determine how many of these can be placed on the chip. For smaller configurations, we determine this by dividing 80% of the total chip area by the processor/cache area. As the number of processors grows, we assume more area overhead (a few additional percent of the total chip area) is required for routing.

Based on this method, we obtain the candidate system organizations shown in Table 1. These provide a good range of design points for an initial analysis. Once we have analyzed these design points, we can make further refinements to arrive at alternative organizations.

Configuration	Number of Processors	Cache Size
1	7	256kB
2	11	128kB
3	16	64kB
4	20	32kB

Table 1: Candidate System Organizations

Cache Size	Miss Rate	Misses Causing Displacements	Displacement Buffer Hits
32kB	2.05%	23.44%	3.96%
64kB	1.33%	27.54%	3.92%
128kB	0.89%	30.85%	4.37%
256kB	0.55%	33.91%	3.74%

Table 2: Cache Simulation Results

We choose to bound the cache sizes at 256kB and 32kB. For our benchmarks, caches larger than 256kB produce diminishing returns in terms of miss rate; further increasing the size of the cache beyond 256kB at the expense of the number of processors clearly results in worse overall performance. Caches of size less than 32kB on the other hand, have high enough miss rates to saturate even a very wide data bus. Even the 256kB and 32kB caches are suspect in terms of these criteria; we include them in order to avoid inadvertently excluding the optimum configuration.

3.3 Modeling Approach

Due to our execution model assumptions, we use uniprocessor trace driven simulation to analyze the various cache options, and apply the results to our multiprocessor analytical models. The traces we use are of the SPEC KENBUS program running on an i486 processor under the MACH 3.0 operating system. These traces are from the BYU Address Collection Hardware (BACH) system[8] and include both user and operating system references. Our trace length is approximately 80 million references, and we ignore cold start effects. The results of our simulations are given in Table 2. Besides miss rate, we also gather information on the fraction of misses that cause a dirty block to be displaced from the cache, and the fraction of cache misses that hit in the four entry displacement buffer. This buffer operates as a FIFO and holds recently displaced blocks to reduce thrashing effects in our direct-mapped caches as described in [11]; our hit rate results are in agreement with this previous work.

The results from Table 2 are used as inputs to analytical models of our candidate organizations. We use Mean-Value-Analysis (MVA), an analytical modeling technique which has been used extensively to study shared memory multiprocessors[4, 18, 19]. Our model assumptions are given in Table 3. CPI_{proc} is the CPI rate of the processor with no cache misses. Our block size choice has been shown in [6] to be a reasonable choice for the KENBUS benchmark for caches in our range. The bus penalties assume that a "dead cycle" is needed to switch bus masters to avoid driver clashing.

Parameter	Value
CPI_{proc}	1.5
% loads	25%
% stores	10%
Block size	32 bytes
Disp buffer hit penalty	2 cycles
Bus read penalty	2 cycles
Bus writeback penalty	3 cycles
Bus data return penalty	3 cycles
Bus arbitration time	2 cycles
Memory access time	15 cycles

Table 3: MVA Model Parameters

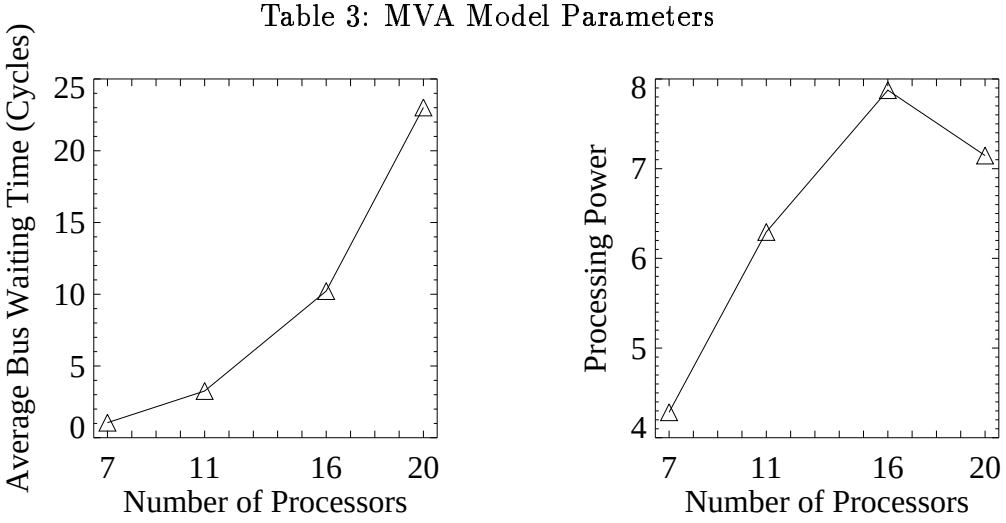

Figure 1: Average Bus Waiting Time and Processing Power for Candidate Configurations

3.4 Processing Power Results

Figure 1 shows the average bus waiting time and processing power for each of the candidate organizations. The larger cache (fewer processor) configurations as expected display very low waiting times, while the waiting time increases quickly for the smaller cache configurations. Thus we conclude that up to a point, adding processors at the expense of cache size is a good tradeoff. We see however that the 20 processor, 32kB cache configuration is clearly not a good design point to consider. The drop in processing power from the 16 processor, 64kB configuration is due to system bus saturation, and the resulting increase in waiting time cancels out the benefits received from adding processors.

The 16 processor, 64kB design point is suspect as well, although it provides the highest processing power. The increase in bus waiting time in this configuration as compared to the 11 processor, 128kB configuration suggests that this design point may not be optimal. It may be beneficial to reduce the number of processors in order to alter the processor organization. One option would be to increase the size of each processor's TLB while reducing the number of processors by an amount commensurate with the resulting area increase. This would result in fewer cache misses and a subsequent reduction in bus waiting time. In order to assess this impact, we use the results of area models developed

Cache Size	Miss Rate	Misses Causing Displacements	Displacement Buffer Hits
32kB	1.72%	24.24%	4.00%
64kB	1.12%	29.50%	4.34%
128kB	0.77%	33.27%	4.68%
256kB	0.49%	35.90%	3.94%

Table 4: Results of Cache Simulations with 512 Entry TLB

for TLBs and caches[14, 15] to assess the relative area of a 32 entry TLB (used in the i486 from which the traces were gathered) and a 64kB cache. From [15], these area estimates are roughly 2500 and 375000 rbes (register bit equivalents), respectively. A 512 entry TLB costs roughly 24000 rbes. Thus, the cost incurred in increasing the TLB from 32 entries to 512 entries is roughly 21500 rbes per processor. For a 15 processor configuration, this amounts to 322500 rbe or less than the cost of one 64kB cache. Thus, we see that by removing one processor and its cache, we can increase the TLB in each of the remaining 15 processors to 512 entries. This should reduce the amount of TLB misses by over 50% from that of a 32 entry TLB[3].

In order to get a lower bound on the impact of this architectural change on performance, we use a technique called *trace modification* to conservatively modify the trace to emulate a trace taken from a processor with a 512 entry TLB. To accomplish this, we use a program that marks the TLB misses in the trace[7]. We then make a conservative estimate as to the number of memory references in the TLB miss code. Our estimate is 35 references, a low estimate based on results of Mach TLB miss behavior[16]. We then insert a filtering program into our simulator that causes the simulator to ignore every other TLB miss in the trace. If a second TLB miss is encountered within 35 references of a filtered TLB miss, we filter it out as well since it results from the first TLB miss. Table 4 shows the results with the larger TLB. Comparing this table with Table 2, we see that for each cache configuration, the miss rate is lower with the larger TLB as expected.

We now use this data to evaluate the impact the larger TLB has on the bus waiting time. We look at various design points for the 64kB cache, with the number of processors ranging from 12 to 16. The larger TLB has a significant effect on the average bus waiting time, reducing it by 17-19%. When viewed from the perspective of equivalent multiprocessor configurations, that is, the number of processors with the larger TLB is one less than that with the 32 entry TLB, the impact is even greater, around a 28% reduction. Looking at these equivalent configurations in terms of processing power (Figure 2), we see that for the smaller configurations, those with the smaller TLB may still provide the best design points. However, we see that the 14 processor, 512 entry TLB and the 15 processor, 32 entry TLB configurations have roughly equivalent performance, and the 15 processor, 512 entry TLB configuration outperforms the 16 processor, 32 entry TLB configuration. Because we have obtained a lower bound on the increase in performance due to enlarging the TLB, the difference in performance may actually be larger than that shown.

3.5 Cycle Time Results

To examine cache access times, we make use of the model developed in [20] which is based on a 0.8 micron, 5V process. We use a voltage of 3.3V for our analysis, and scale the capacitance, resistance, and current parameters of this model to produce a 0.25

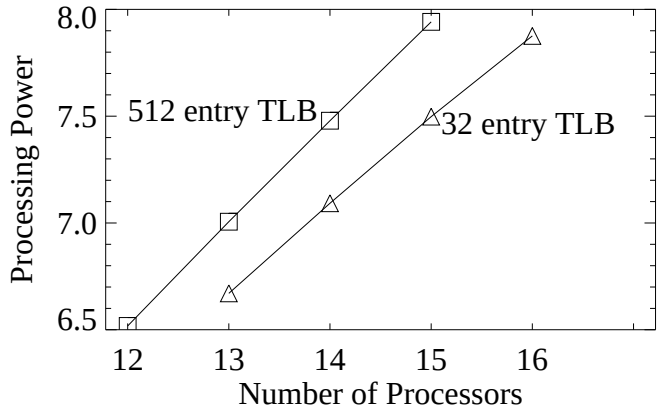

Figure 2: Processing Power for Equivalent 64kB Configurations

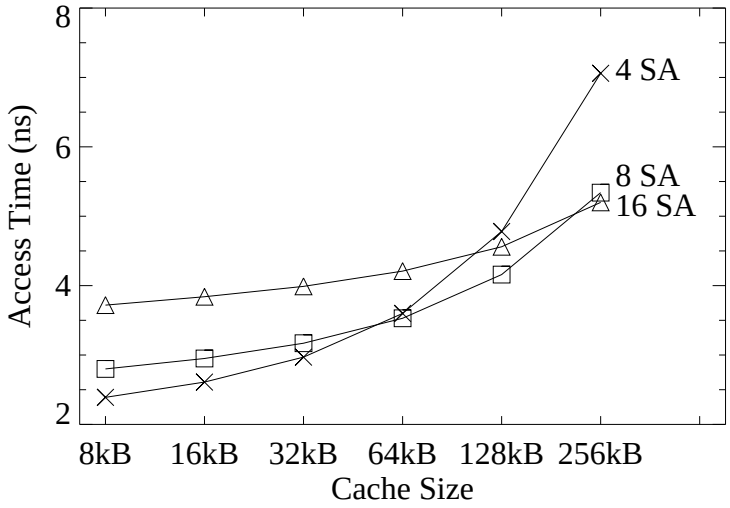

Figure 3: Cache Access Times for Various Sizes and Geometries

micron technology model[2]. We examine 4 (the same as the 21064), 8, and 16 subarray (SA) caches. Our results (Figure 3) indicate that caches larger than 64kB require more aggressive geometric design to achieve reasonable cycle times. This may result in more area overhead and less layout flexibility.

We now examine the effects of the number of processor nodes on bus performance. We assume equal space loading and since the total propagation delay is longer than the driver rise time, we use transmission line analysis. The loaded propagation delay of a transmission line is[2] $T_0 \cdot \sqrt{1 + C_D/C_0}$ where T_0 and C_0 are the unloaded transmission line propagation delay and capacitance, respectively, and C_D is the distributed capacitance due to each processor node. C_D can be expressed as $C_N \cdot s/l$ where C_N is the node capacitance, s is the number of line segments (one less than the number of nodes), and l is the length of the bus (20mm in our example). Following [2], we use C_N = 5fF, C_0 = 2pF/cm, and T_0 = 0.067ns/cm, and obtain the results shown in Figure 4. We conclude that shared buses can still achieve good cycle time performance for moderate numbers of processors. However, to achieve aggressive clock rates with large numbers of processors,

[2]We note that these models are highly dependent on technology parameters, and thus our method should only be used to study general trends and not exact cycle time analysis.

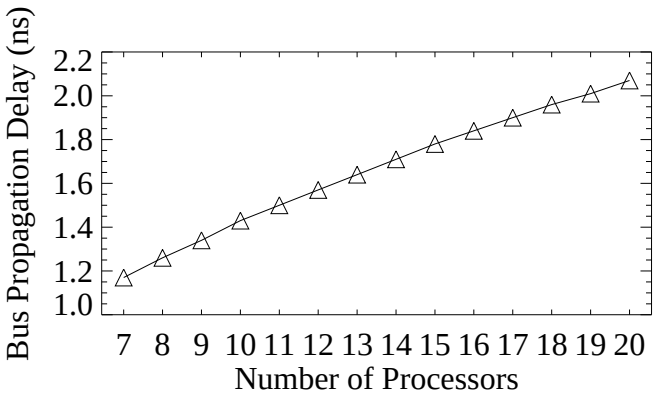

Figure 4: Bus Propagation Delay Variation with Number of Processors

Configuration	PP	Cycle Time		Total System Performance	
(Proc, Cache, TLB)		Cache Constrained	Bus Constrained	Cache Constrained	Bus Constrained
7, 256kB, 32 entry	1.00	1.00	1.00	1.00	1.00
11, 128kB, 32 entry	1.47	0.80	1.28	1.84	1.15
15, 64kB, 512 entry	1.85	0.68	1.52	2.72	1.22
16, 64kB, 32 entry	1.84	0.68	1.57	2.71	1.17

Table 5: Performance Data for Candidate Configurations

interconnect alternatives such as ring-based schemes[9] need to be considered.

3.6 Overall Performance

We now bring together the processing power and cycle time results and compare the overall performance of our candidate configurations. We use the 7 processor, 256kB configuration as our base system (and thus assign it unity performance values), and calculate processing power, cycle time, and total system performance for the other configurations relative to this organization. We look at two different cycle time scenarios: a *cache constrained* cycle time chip, and a *bus constrained* cycle time chip. In the former, the cache access time determines the cycle time of the chip; in the latter, the bus propagation delay determines this.

Table 5 shows the relative processing power (PP), cycle time, and total system performance for each of the candidate configurations. Due to their superior processing power performance, the 15 and 16 processor configurations achieve the best overall performance in both the cache and bus constrained cases. The 15 processor configuration is overall the best choice, as it provides the highest processing power, and an equal or faster cycle time than the 16 processor configuration. We note, however, that for a bus constrained cycle time chip, the performance of the 11 processor configuration closely matches that of the 15 and 16 processor configurations. For applications that consume more of the cache, this configuration may be the best choice. If an even number of processors (or a power of two) is more beneficial for the application software, then the number of processors can be reduced accordingly. This may require the designer to allocate more area to TLBs, interconnect, or other resources.

4 Conclusions and Future Work

The complexity of microprocessor design is growing rapidly as VLSI integration levels continue to increase. With the technology expected to be available at the end of this decade, microprocessor architects will be faced with a wide range of design decisions. An analysis of tradeoffs between the size of the caches and the number of processors in the system was presented. Our results show that trading off cache size for the number of processors improves performance up to the point of bus saturation. At this point, alternatives such as reducing the number of processors in order to increase TLB size, need to be considered in order to arrive at the optimal design point.

This work can be extended in several different ways. First of all, we assumed an independent execution model, whereby processors are accessing private code and data and negligible sharing takes place. We also looked at a single benchmark program. Examining the effect of a wider range of execution models (e.g., fine-grain parallel processing) and application programs on the design choices made would be insightful. Secondly, an examination of the sensitivity of our results to our model parameters (such as memory access time) should be made. Lastly, we only considered a single level of cache hierarchy. Multilevel cache hierarchies can be examined as well. The incorporation of multiple processors on a chip allows the designer to consider other cache options, such as private first level caches and multiported second level caches shared by two or more processors. Such an arrangement may make more efficient use of the available chip area than a conventional organization, and thus should be evaluated as well.

Acknowledgements

The authors wish to thank the members of the BACH project at Brigham Young University, especially Kelly Flanagan, Knut Grimsrud, and Brent Nelson, for providing the traces and the TLB miss marking tool used in this project. We also thank the anonymous referees for their helpful comments and suggestions.

References

[1] H.B. Bakoglu and J.D. Meindl, "Optimal Interconnection Circuits for VLSI," *IEEE Transactions on Electron Devices*, pp. 903-909, May 1985.

[2] H.B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.

[3] J.B Chen, A. Borg, and N.P. Jouppi, "A Simulation Based Study of TLB Performance," *19th International Symposium on Computer Architecture*, pp. 114-123, May 1992.

[4] M. Chiang and G.S. Sohi, "Evaluating Design Choices for Shared Bus Multiprocessors in a Throughput-Oriented Environment," *IEEE Transactions on Computers*, pp. 297-317, March 1992.

[5] D.P. Dobberpuhl, et al, "A 200MHz, 64-Bit, Dual-Issue CMOS Microprocessor," *Digital Technical Journal*, Vol. 4, No. 4, pp. 35-50, 1992.

[6] J.K. Flanagan, "A New Methodology for Accurate Trace Collection and Its Application to Memory Hierarchy Performance Modeling," PhD Dissertation, Brigham Young University, December 1993.

[7] K. Grimsrud, "Address Translation of BACH i486 Traces," Technical Memorandum, Brigham Young University, June 1993.

[8] K. Grimsrud, J. Archibald, M. Ripley, K. Flanagan, and B. Nelson, "BACH: A Hardware Monitor for Tracing Microprocessor-Based Systems," Technical Report, Brigham Young University, February 1993.

[9] D. Gustavson, "The Scalable Coherent Interface and Related Standards Projects," *IEEE Micro*, pp. 10-22, February 1992.

[10] M. Hanawa, et al, "On-Chip Multiple Superscalar Processors with Secondary Cache Memories," *International Conference on Computer Design*, pp. 128-131, October 1991.

[11] N.P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," *17th International Symposium on Computer Architecture*, pp. 364-373, May 1990.

[12] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik, *Quantative System Performance, Computer Analysis Using Queuing Network Models*, Prentice Hall, Englewood Cliffs, N.J., 1991.

[13] E. McLellan, "The Alpha AXP Architecture and 21064 Processor," *IEEE Micro*, pp. 36-47, June 1993.

[14] J.M. Mulder, N.T. Quach, M.J. Flynn, "An Area Model for On-Chip Memories and Its Application," *IEEE Journal of Solid-State Circuits*, pp. 98-106, February 1991.

[15] D. Nagle, R. Uhlig, T. Mudge, and S. Sechrest, "Optimal Allocation of On-chip Memory for Multiple-API Operating Systems," *21st International Symposium on Computer Architecture*, April 1994.

[16] D. Nagle, R. Uhlig, T. Stanley, S. Sechrest, T. Mudge, and Richard Brown, "Design Tradeoffs for Software-Managed TLBs," *20th International Symposium on Computer Architecture*, pp. 27-38, May 1993.

[17] J. Rattner, "MICRO 2000: Microprocessors in the Year 2000," Keynote Address, Second Annual VLSI Educator's Conference, July 1989.

[18] M.K. Vernon, R. Jog, and G.S. Sohi, "Performance Analysis of Hierarchical Cache-Consistent Multiprocessors," *Performance Evaluation*, Vol. 9, pp, 287-302, 1989.

[19] M.K. Vernon, E.D. Lazowska, and J. Zahorjan, "An Accurate and Efficient Performance Analysis Technique for Multiprocessor Snooping Cache Consistency Protocols," *15th International Symposium on Computer Architecture*, pp. 308-315, June 1988.

[20] T. Wada, S. Rajan, S.A. Przybylski, "An Analytical Access Time Model for On-Chip Cache Memories," *IEEE Journal of Solid-State Circuits*, pp. 1147-1156, August 1992.