STATS: A Framework for Microprocessor and System-Level Design Space Exploration

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

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

Keywords: microprocessor design, performance evaluation, memory hierarchies, superscalar processors

Abstract

As microprocessor-based systems grow in complexity, and the processor-memory speed gap widens further, more emphasis needs to be placed on early design space exploration in order to produce the highest performance systems with minimal schedule impact. We discuss the critical issues associated with architectural evaluation of complex microprocessor-based systems, and present a methodology for the comprehensive and semi-automatic evaluation of processor, cache hierarchy, system interconnect, and main memory architectural and technological alternatives. We discuss the implementation of the methodology, and describe how it can be used in early design space exploration. The unique aspects of the methodology are further illustrated through two architectural investigations performed using the toolset.

1 Introduction

The design of commercial microprocessor computer systems typically begins with the comparative evaluation of architectural alternatives, resulting in architectural specifications to guide subsequent design phases. This procedure plays an important role in computer architecture research as well in determining the merits of new ideas. Simulation is commonly the vehicle for architectural analysis due to its simpler resource requirements and greater flexibility relative to prototyping, and (in many cases) higher accuracy relative to analytical modeling.

Although simulation methods have produced significant advances in commercial systems development and architecture research, three architectural and technological trends threaten to limit the applicability of current approaches for designing next-generation microprocessor systems. The first is that due to increasing levels of VLSI integration fueling greater opportunities for architectural innovation, the complexity of microprocessor design is increasing rapidly. Microprocessor designers are faced with more decisions in the early design phase than ever before, and the number and complexity of these decisions is growing rapidly with each new microprocessor generation. To wade through these various options requires that the architect be equipped with highly-parameterized tools that automate the analysis process while providing extensive performance feedback information to identify bottlenecks.

The second trend is that performance is becoming more dependent on system-level design due to the widening speed gap between microprocessor and system-level technology. As microprocessor clock rates continue to climb, delays through microprocessor packages, printed circuit board traces, and board-level components constitute a larger fraction of expended microprocessor cycles, and thus have a greater impact on overall performance. Therefore, it is crucial for architects to include these details in evaluating alternatives in order to make informed architectural tradeoffs.

The third is the use of microprocessors in a wide range of *target systems*, such as personal computers, workstations, and multiprocessor servers, which may vary radically in system-level organization and technology. This calls for the architect to assess the performance of each candidate microprocessor configuration using a variety of system-level models to determine its overall suitability.

The **S**ystem **T**radeoff **A**nalysis **T**oolset (STATS) is a highly parameterized, flexible framework for the early design space exploration of high performance microarchitectures. STATS encompasses a number of features not found in other analysis tools including simultaneous evaluation of microprocessor and system-level design alternatives, the inclusion of microarchitectural, technological, and topological considerations within a single framework, the use of result databases to speed design space exploration, and the enabling of automated design space searches. The result is a flexible tool incorporating a high degree of design detail that can be used by an interdisciplinary design team to explore alternatives in the early, high-impact, architectural definition stage of the development process.

The rest of this paper describes STATS in detail including its features, implementation, and limitations, and demonstrates its applicability to the design of high performance microarchitectures.

2 Related Research

Simulation-based computer architectural analysis has been a rich area of research, most of which has focused on the design of a single subsystem such as the processor (*e.g.*, [10, 12, 21]) or cache hierarchy (*e.g.*, [11, 18, 19, 24, 28]). Most of these studies focus on a small portion of the design space, and only a small subset take technology and implementation constraints into account in the analysis. Olukotun's studies of primary cache design for a multi-chip module (MCM) based Gallium Arsenide microprocessor [18, 19] include a linear equation for calculating the delay between the processor and primary cache chips on the MCM. Jouppi and Wilton [11] use a detailed cycle time model in addition to simulation to study the performance of two-level on-chip caching relative to a single level of on-chip cache. Uhlig [24] includes on-chip implementation-related parameters such as latency and bandwidth in analyzing various two-level on-chip cache structures. Values for these parameters are obtained from existing processor designs, or a range of latency and bandwidth values is explored.

Some architectural evaluation tools and studies have attempted to span a wider scope of machine architecture. For example, the PowerPC performance simulation tool [20] allows the architect to simultaneously evaluate processor, cache, and bus design tradeoffs[1]. VMW [4] is an architectural simulator that is designed to include simulator retargetability, visualization of results, and interactive control in performance analysis, and encompasses multiple subsystems as well. A cost/performance analysis of the design of an experimental 300MHz microprocessor using Gallium Arsenide technology [25] includes a number of processor hardware parameters, including instruction and data cache sizes and number of execution pipelines.

Although these techniques have been used with much success in evaluating architectural alternatives, there remains the need to provide a more comprehensive approach that allows a wide range of options within multiple subsystems to be simultaneously evaluated, and which incorporates implementation constraints and multiple target system designs into the methodology. The remainder of this paper describes how STATS builds on the ideas of this earlier work to more completely address the trends discussed in Section 1.

[1] Interestingly, the author states that in the future, microprocessor designers should work more closely up-front with system designers to achieve better overall performance.

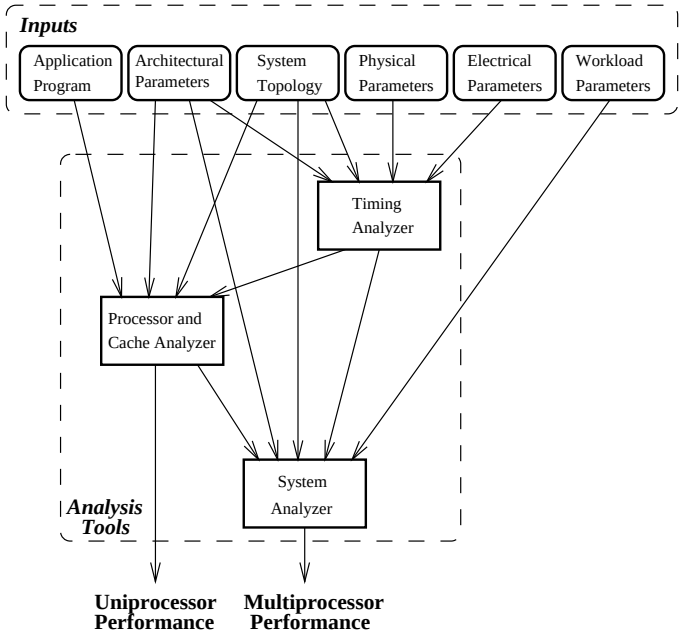

Figure 1: High-level organization of STATS.

3 Organization of STATS

Figure 1 shows the overall STATS organization. Arrows indicate the flow of input parameters and intermediate results between the various analysis tools. Results required by these tools are defined by input templates read by Perl scripts that direct the flow of inputs and results during analysis. This modular structure facilitates expansion and customization of the toolset, as well as automated design space exploration as will be described later in Section 4.

The toolset currently encompasses over 800 architecture, technology, and system topology input parameters covering a wide design space. Architectural parameters cover the processor, cache hierarchy, system interconnect, and main memory subsystems[2]. Technology parameters (incorporated within Physical and Electrical Parameters in Figure 1) cover microprocessor and system-level ASIC VLSI technology, packaging at the chip, daughtercard, board, and backplane levels, and SRAM and DRAM specifications. System topology parameters describe the overall connection of the various system components, and include a single-board configuration, a daughtercard-motherboard arrangement, or a backplane in which multiple boards are plugged. This flexibility allows the architect to experiment with tradeoffs with different types of packaging, and to model a number of uniprocessor and multiprocessor systems.

Although the prospect of specifying 800 parameters

[2] The I/O subsystem is currently only modeled as one or more ASICs loading the external bus.

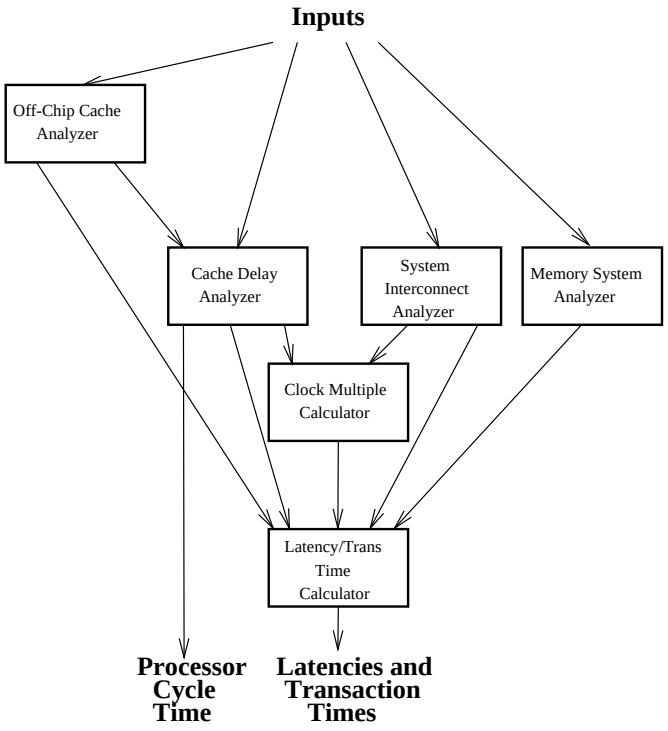

Figure 2: Timing Analyzer organization.

may appear daunting, it forces the architect to realistically account for technology and implementation constraints by consulting technology experts and/or specifications early in the design process, instead of relying on guesswork or possibly outdated values from previous designs. The result is a more accurate assessment of the performance of candidate configurations, with fewer surprises in the later implementation phases of the design process. In addition, the bulk of these parameters are either architectural parameters, or parameters that need only be specified initially, considerably simplifying this process.

At the heart of STATS are three main analysis tools: the Timing Analyzer, the Processor and Cache Analyzer, and the System Analyzer, which are described in the following subsections.

3.1 Timing Analyzer

An important feature of STATS is the integration of timing analysis with very detailed microarchitectural simulation and architectural-driven benchmark compilation. The Timing Analyzer (Figure 2) estimates the cycle time of the processor and the latencies and transaction times of the cache hierarchy, system interconnect, and main memory. The Off-Chip Cache Analyzer determines the address and data bus delays of off-chip SRAM arrays (both asynchronous and synchronous RAMs are supported), adding in designer-provided specifications for clock to output delay, setup time, and clock skew. This information is passed to the Cache Delay Analyzer, which evaluates the critical cycle time path delays within the cache hierarchy, using an enhanced version of CACTI [29] to estimate on-chip cache delays. The enhanced version models pipelined caches by optimally inserting registers between the various major substages of the cache. The substages include data and tag decode, data and tag array access, data out drive, tag compare, multiplexer drive, and data select. The last two stages are for a set-associative cache and are replaced by a single valid out signal stage for a direct mapped cache. Up to 4 data stages, and 5 (direct mapped) or 6 (set associative) tag stages are allowed. For caches specified to have less than the maximum number of stages, the model finds the optimal register placement in terms of minimizing cycle time. The cycle time is the largest of the access time plus precharge time of the tag array, access time plus precharge time of the data array, and the register-register timing of each stage. For modeling multi-ported caches, the designer inputs the number of duplicate arrays per cache and ports per array as well as a scaling factor that corresponds to the increase in cell area incurred with each added port. This factor is used to scale the size of the cell and resulting word and bitline capacitances as well as the increased cell capacitance due to the presence of multiple output port transistors.

Worst case delays of the external system bus, and of control and data paths of main memory DRAM-based arrays are estimated by the System Interconnect Analyzer and the Main Memory Analyzer, respectively. These tools and the Off-Chip Cache Analyzer use the automated Spice-based methodology shown in Figure 3. Spice timing path models are constructed from input parameters, including detailed technology specifications, by the Spice Model Builder, and results are extracted from the Spice Simulation Database if this configuration has been previously analyzed. Otherwise, Spice runs are performed (both rise/fall transitions using min/max transition times), the worst-case delay determined by the Output Extractor from the result files, and the database updated. The use of Spice ensures that transmission line effects, such as reflections, are properly accounted for, and the output results are stored so that the designer can view them graphically if desired. For buses with more than one source (such as the system bus) the designer has the option of analyzing all possible sources or only characterizing the longest end to end path, allowing the designer to trade off a more comprehensive analysis for greater analysis speed.

The Clock Multiple Calculator determines the clock multiplier factor that the external bus and main memory subsystems will operate at by dividing the ex-

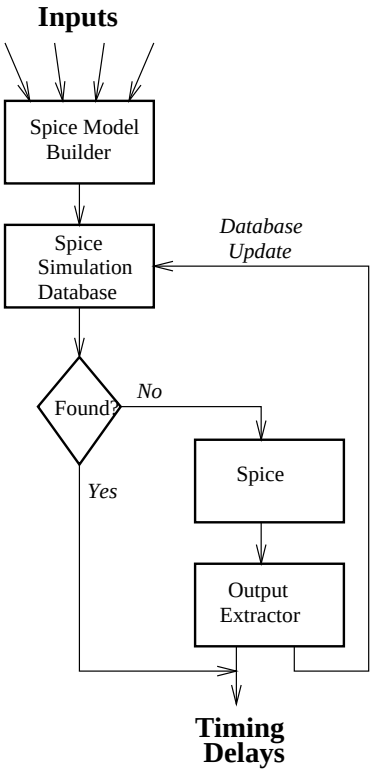

Figure 3: Automated Spice-based timing tool.

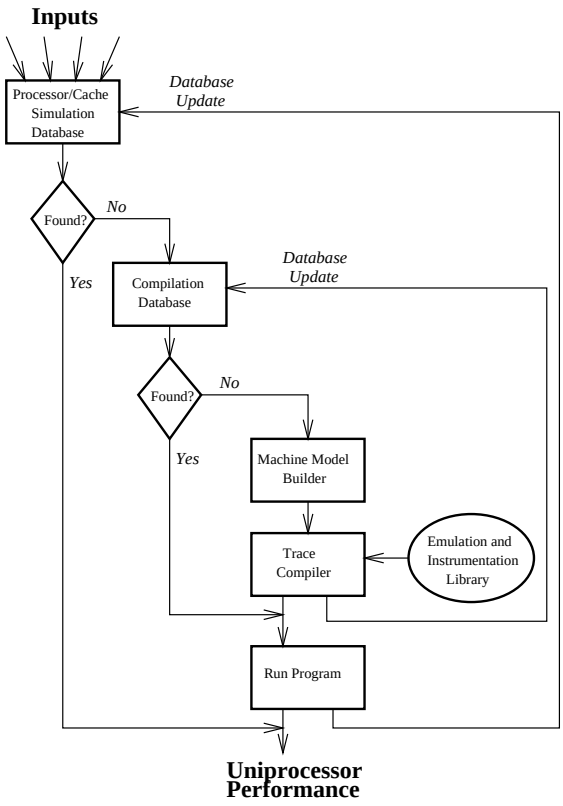

Figure 4: Processor and Cache Analyzer organization.

ternal bus latency by the processor cycle time, and rounding up to the next highest integer. The Latency/Transaction Time Calculator then determines latencies and transaction times for the cache hierarchy, system interconnect, and main memory subsystems. Timings for the latter two subsystems are calculated in terms of the external interface clock multiple. Thus, the granularity of system-level operations, such as system bus and main memory transactions, is impacted by the relative speeds of the on-chip and off-chip clocks.

3.2 Processor and Cache Analyzer

The Timing Analyzer estimates are used as inputs to the Processor and Cache Analyzer (Figure 4), an integrated microarchitectural compilation and simulation analysis environment which like the Timing Analyzer, uses databases to avoid duplication of effort to speed design space exploration. If the configuration has been previously analyzed, then the results are returned by the Processor/Cache Simulation Database. Otherwise, a lookup is done in the Compilation Database. A compilation is required if the application program has not been previously compiled or the processor organization (including issue width, number and types of functional units, latencies for each instruction type, and producer-consumer latencies) has not been previously analyzed. (Cache and other parameters are provided as inputs to the compiled program.) If a compile is required, the Machine Model Builder organizes these processor-specific parameters in the format required by the Trace Compiler, a version of the Multiflow Trace Scheduling Compiler [13] targetted to the Digital Alpha architecture. The compiler uses this information to generate code optimized for the processor architecture, ensuring that individually optimized benchmark executables are generated for each machine configuration.

Linked with the benchmark object file are library routines within the Emulation and Instrumentation Library that instrument the code to measure static cycle counts (e.g., due to functional unit latencies) and model dynamic behavior (e.g., due to branches and cache accesses). A large amount of flexibility has been built into these library routines. For example, the cache hierarchy routines allow all important organizational parameters (size, associativity, block size, *etc.*) to be varied. Support for nonblocking caches, write and victim buffers, and one, two, or three levels of cache hierarchy is also included. Execution of the program invokes parameter input and emulates the target machine. The resulting cycle count information is combined with cycle time estimates from the Timing Analyzer to predict uniprocessor performance for

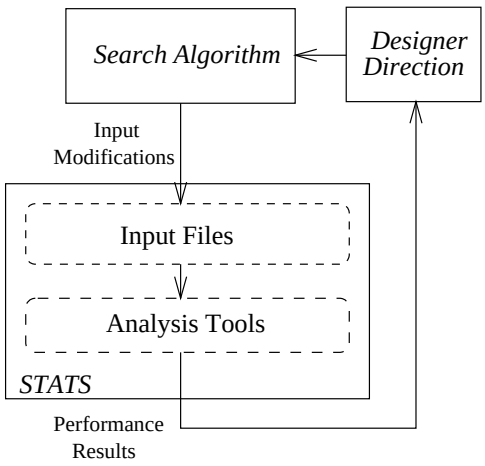

Figure 5: Using STATS to automatically search the design space.

each configuration. Detailed performance information is saved to allow the architect to analyze bottlenecks and direct the search process.

3.3 System Analyzer

The System Analyzer is a detailed Mean Value Analysis (MVA) model of the System Interconnect and Main Memory subsystems. The model uses latency and transaction time values from the Timing Analyzer, and reference rate information from the Processor and Cache Analyzer to compute multiprocessor system performance. Multiprocessor workload parameters (such as the probability that on a load miss the desired block is in a "modified" state in another cache) are provided by the designer. MVA modeling of shared memory multiprocessors is described by Vernon [26] and Jog [9] among others.

4 Design Space Exploration with STATS

The integrated nature of STATS and the text file input parameter interface facilitates automated design space exploration as is shown in Figure 5. Here, a search algorithm is used to modify STATS's input parameter files and call STATS for each candidate configuration. For example, an exhaustive search of a portion of the design space can be performed by creating a simple script file that modifies input parameters and calls STATS for each configuration, using a nested loop for each parameter to be explored. A more complex search may be performed using a genetic algorithm [23] to prune the search tree, and to dispatch work among a network of workstations [22], each of which runs STATS on a different candidate configuration. In the next section, we describe two architectural investigations we have performed using STATS and a simple exhaustive search approach.

Levels	Size			
	L1I	L1D	L2	L3
1	256KB	256KB	—	—
1	512KB	512KB	—	—
1	1MB	1MB	—	—
2	8KB	8KB	256KB	—
2	8KB	8KB	512KB	—
2	8KB	8KB	1MB	—
2	8KB	8KB	2MB	—
2	16KB	16KB	256KB	—
2	16KB	16KB	512KB	—
2	16KB	16KB	1MB	—
2	16KB	16KB	2MB	—
2	32KB	32KB	256KB	—
2	32KB	32KB	512KB	—
2	32KB	32KB	1MB	—
2	32KB	32KB	2MB	—
3	8KB	8KB	96KB	256KB
3	8KB	8KB	96KB	512KB
3	8KB	8KB	96KB	1MB
3	8KB	8KB	96KB	2MB
3	8KB	8KB	128KB	256KB
3	8KB	8KB	128KB	512KB
3	8KB	8KB	128KB	1MB
3	8KB	8KB	128KB	2MB

Table 1: Candidate Cache Configurations

5 Applications of STATS

We describe in this section the application of STATS to superscalar processor and memory hierarchy design, and demonstrate the coverage of multiple subsystems and the impact of technology constraints on architectural tradeoffs. Because we use the SPEC92 benchmarks, which are limited in their effectiveness in analyzing cache behavior [7], our analysis is intended to serve as an example of the capabilities of the toolset rather than an endorsement of particular design options. For space reasons, we provide only a sampling of the results from these studies. A complete description of our assumptions and results, and other tradeoffs we have investigated is provided in [1, 2].

5.1 Technology Considerations in Single and Multi-Level Cache Design

We use STATS to investigate the impact of technology considerations and the number of levels in cache hierarchy design. The processor pipeline is a two-way superscalar design similar to that of the Alpha 21064A microprocessor [5]. We consider 23 different cache topologies (Table 1) of various sizes and which incorporate between one and three levels of hierarchy. Each option uses a board-level cache, whose timings are precisely characterized using the Timing Analyzer and actual SRAM specifications, enabling off-chip cache latency and bandwidth effects to be accurately accounted for. Seven board-level cache configurations are examined using three types of fast asynchronous SRAMs (32K×8 [15], 64K×4 [16], and

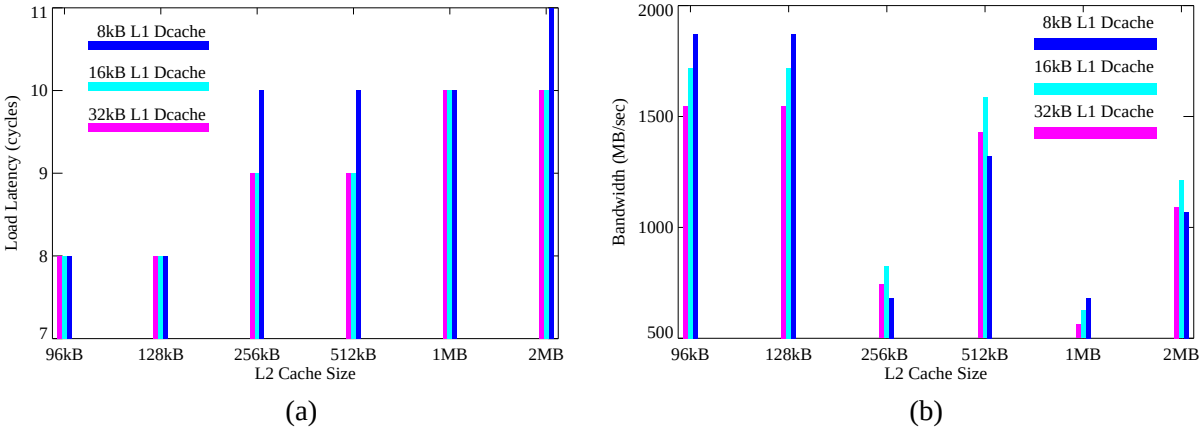

Figure 6: L2 Cache (a) Load Latency and (b) Bandwidth

128K×8 [17]). For two and three-level configurations, we use a total of either 9 or 18 of the 32K×8 or 128K×8 SRAMs for the data array, and assess the relative performance of these different cost and bandwidth alternatives together with the various on-chip cache configurations[3].

Figure 6 shows how load latency and bandwidth vary for on (96KB and 128KB) and off-chip (256KB to 2MB) L2 caches and how this is impacted by the size of the L1 Dcache. Due to their smaller size and lower delays, the on-chip L2 caches have considerably lower load latency and higher bandwidth than their off-chip counterparts. The general increase in load latency observed for the off-chip L2 caches is due to the use of more and/or denser (and slower) SRAMs to construct larger caches. For the 256KB, 512KB, and 2MB L2 caches, load latency increases at the 8KB L1 Dcache points because the smaller cycle time at this design point increases the number of cycles required to access the L2 cache. Similar variations are observed with bandwidth (Figure 6(b)). These intuitively second-order effects have significant performance ramifications for some benchmarks. Figure 7 shows the performance in instructions/second of the various cache alternatives for the *su2cor* benchmark. The performance difference between two-level configurations with 8KB and 16KB L1 Dcaches is 15% with a 1MB L2 cache, and jumps to 30% with a 2MB L2 cache due to these effects. In addition, a pronounced dip is observed for two-level configurations at the lower-bandwidth 1MB L2 cache design point. The absence of a dip for the three-level curves is due to the lower bandwidth requirement imposed on an L3 cache

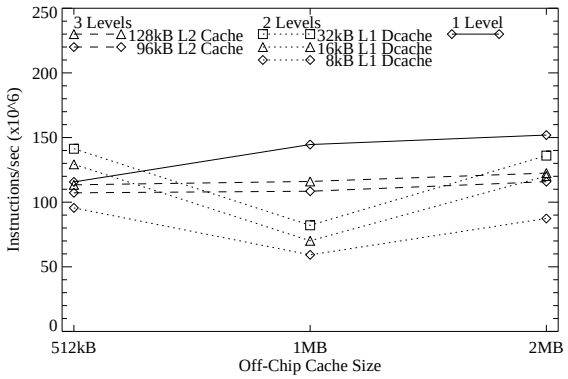

Figure 7: Performance of *su2cor* for Each Cache Hierarchy Option.

due to the presence of the on-chip L2 cache[4]. Thus, by using a two-level on-chip cache, system design costs may potentially be reduced by requiring fewer expensive SRAM components and a smaller microprocessor package due to this lower board-level cache bandwidth requirement.

5.2 Multiple Subsystem Design Tradeoffs

As a second example, we consider the design of an 8-way in-order superscalar engine modeled after the 21164 microprocessor [6] used in a single processor workstation, and examine trading off the number of arithmetic functional units for L1 Dcache size, three L1 Dcache multi-porting alternatives, and two system topology options (Figures 8-10). Figure 8 shows two area-equivalent on-chip options: one with 12 arith-

[3]The bandwidth variations are due to differences in data bus width, SRAM access times, and SRAM package dimensions (which impacts the length of the address and data buses).

[4]The single-level caches also don't have this limitation as their data width is set at 64 bits to match the width of the 21064A's single load/store unit.

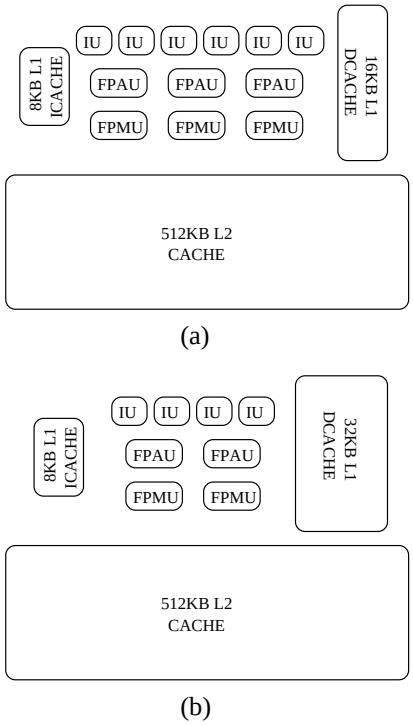

Figure 8: FU and L1 Dcache Options: (a) 12 FUs, 16KB L1 Dcache, (b) 8 FUs, 32KB L1 Dcache

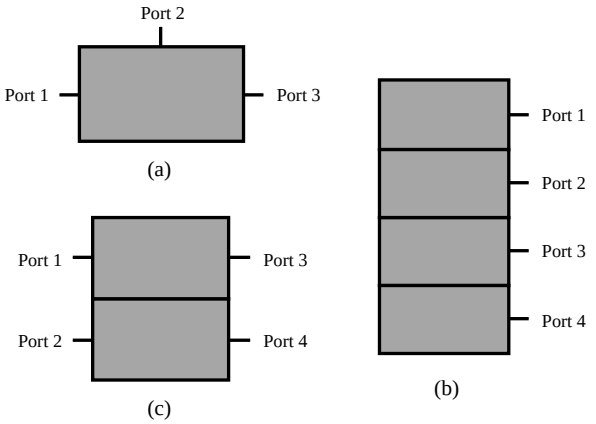

Figure 9: L1 Dcache Multi-Porting Options: (a) 1TP, (b) 4SP, (c) 2DP.

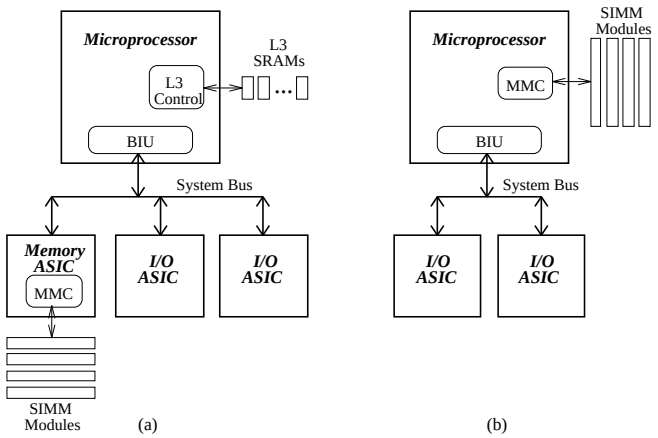

Figure 10: System Topology Options: (a) L3 Cache and (b) Main Memory.

metic functional units (FUs) consisting of 6 integer units (IUs) and three floating-point add (FPAU) and three floating-point multiply (FPMU) units coupled with a 16KB L1 Dcache, and another with a total of 8 FUs and a 32KB L1 Dcache. As an 8-way issue processor demands high bandwidth L1 Dcache access, we explore three multi-porting alternatives shown in Figure 9 which vary in cycle time and port-use flexibility: a single triple-ported array (1TP), four single-ported arrays (4SP), and two dual-ported arrays (2DP). Finally, Figure 10 illustrates the two memory hierarchy and system topology options that we explore: the use of package pins for a 4MB L3 cache (with the Main Memory Controller (MMC) implemented in a separate ASIC), and a direct connection to Main Memory SIMM modules [14] (no L3 cache).

Figure 11 shows how performance varies from benchmark to benchmark with the various L1 Dcache multi-porting options. Processor elapsed time is plotted for the 4SP and 2DP caches, normalized with respect to the elapsed time of the 1TP cache organization. (Figure 11(a) is normalized to the 8 FU configuration, and Figure 11(b) to the 12 FU configuration.) Both the 2DP and 4SP configurations execute all benchmarks faster than the 1TP design. For four of the benchmarks, the cycle time advantage of the 4SP cache overrides the port-use flexibility of the 2DP cache. Nevertheless, the 2DP scheme outperforms the 4SP design overall, and performs markedly better with workloads with high L1 Dcache load and store activity (*li* and *tomcatv*), for which the compiler scheduler is less constrained by cache port limitations. The single store per cycle limitation of the 4SP cache is a particularly severe bottleneck for *tomcatv*, for which there is a significant performance improvement with the more balanced 2DP design. In addition, the 2DP design requires 15% less area than the 4SP configuration [1].

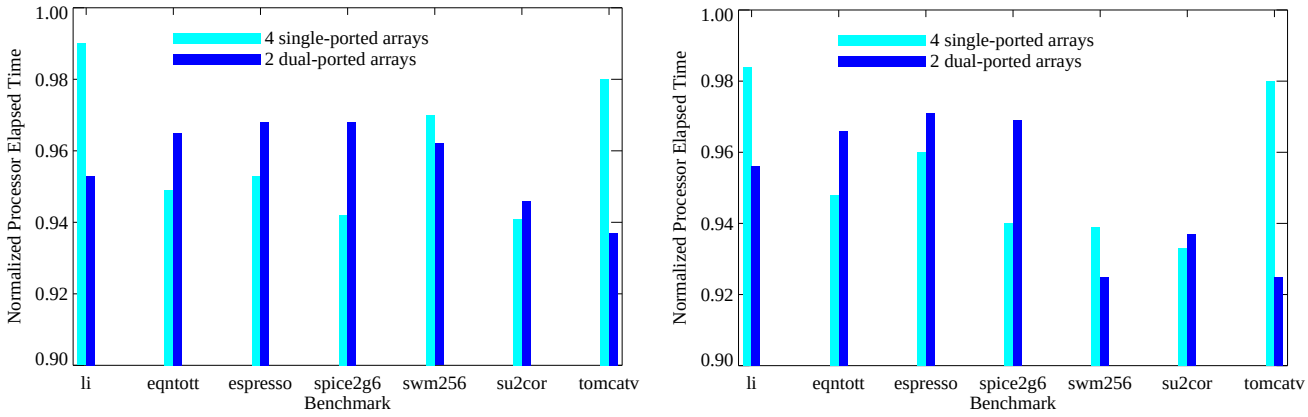

Figure 11: Normalized Processor Time for Multi-Porting Options with (a) 8 FUs, 32KB L1 Dcache and (b) 12 FUs, 16KB L1 Dcache.

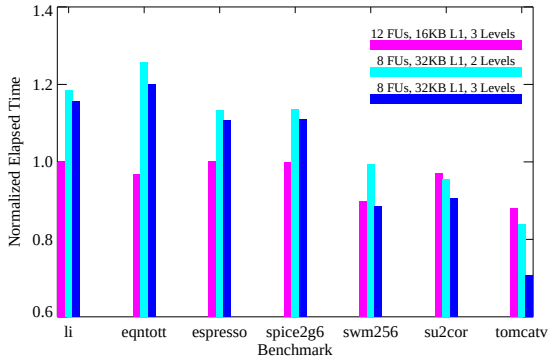

Figure 12: Elapsed Time of Three Configurations Normalized to the 12 FU, 16KB L1 Dcache, 2-Level Configuration.

Figure 12 demonstrates the performance tradeoffs of functional units versus L1 Dcache size and the two system topology options (two levels of cache with an integrated MMC versus three levels with a separate MMC ASIC). Results are normalized to the 12 FU, 2-level configuration. With all benchmarks given equal weight, the 3-level cache organizations require marginally less total time (2% and 4% for the 8 and 12 FU configurations, respectively) than the 2-level organizations, although the difference is substantial for *swm256* and *tomcatv*, two of the most cache-intensive benchmarks.

Overall, the 12 FU, 3-level cache configuration outperforms the 8 FU, 3-level cache configuration by 7%, indicating that increasing L1 Dcache cache size at the expense of functional units does not yield best performance. However, three of the floating point benchmarks (*swm256*, *su2cor*, and *tomcatv*) perform best with the larger L1 Dcache, demonstrating the expected application dependency on this tradeoff. The main point is that STATS provides the framework for making these types of tradeoffs with very little additional work required of the architect compared with less comprehensive approaches.

6 Conclusions and Future Development

In this paper, we have described STATS, an architectural evaluation framework developed to address the problems of searching the ever-increasing design space of microprocessors, accurately modeling technology and implementation constraints in architectural evaluation, and characterizing candidate microprocessor designs using multiple target system models. We have demonstrated how STATS can be used to take SRAM and packaging technology limitations into account in cache hierarchy design, and to simultaneously explore processor, cache hierarchy, and system topology options in microprocessor systems design.

STATS currently contains several limitations which need to be addressed in future versions. First, the cycle time model assumes that the critical path lies within the cache hierarchy. Although this may be true in many machines, highly parallel architectures exploiting instruction level parallelism may severely increase processor timing paths [8], calling for additional pipeline stages and/or an increase in cycle time. Accurate characterization of these timings requires more effort than with caches due to the prevalence of complex control logic. However, performing detailed design of the perceived processor critical timing paths for a multitude of processor options inhibits the objective of rapidly covering a broad design space. The goal is

to find the middle ground between accuracy and effort, by creating parameterized critical timing path models representative of general processor designs from which reasonably accurate cycle time and latency estimates can be quickly produced.

Second, the Multiflow compiler and Alpha code generator restrict the types of compiler optimizations that can be performed as well as the instruction set architectures supported. Modifying STATS to use the SUIF compiler [27], which is publicly available, supports modern compiler optimizations, and has been ported to most major platforms, would remove both of these limitations. In addition, SUIF supports parallelization of sequential code for multiprocessor systems [3], which would help improve the modeling of finer grain parallel processing than is currently represented in STATS.

Finally, STATS is constrained by an in-order issuing model, limited branch prediction mechanisms, and a system bus as the only interconnect option, all of which will be addressed in later versions of the toolset. STATS will need to be continually upgraded to reflect the current state-of-the-art in microprocessor and system-level design.

Acknowledgements

The authors thank Tryggve Fossum, Michael Adler, Joel Emer, Geoff Lowney, Bob Nix, and David Webb of Digital Equipment Corporation for developing and licensing the compilation and simulation infrastructure that we adapted for STATS. We also thank Norman Jouppi and Steve Wilton for making CACTI available, and to Steve for explaining its intricacies.

References

[1] D.H. Albonesi. *Architecture and Technology Tradeoffs in the Design of High Performance Microprocessor-Based Systems*. PhD thesis, University of Massachusetts, Amherst, MA, September 1996.

[2] D.H. Albonesi and I. Koren. Architecture and technology tradeoffs in the design of next-generation multiprocessor servers. *Proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing*, pages 174–181, October 1995.

[3] S.P. Amarasinghe et al. Multiprocessors from a software perspective. *IEEE Micro*, 16(3):52–61, June 1996.

[4] T.A. Diep and J.P. Shen. VMW: A visualization-based microarchitecture workbench. *IEEE Computer*, 28(12):57–64, December 1995.

[5] Digital Equipment Corporation. Alpha 21064A microprocessor product brief. March 1995.

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

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

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

[9] R. Jog, P.L. Vitale, and J.R. Callister. Performance evaluation of a commercial cache-coherent shared memory multiprocessor. *Proceedings of SIGMETRICS*, pages 173–182, May 1990.

[10] W.M. Johnson. *Superscalar Microprocessor Design*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1991.

[11] N.P. Jouppi and S.J.E. Wilton. Tradeoffs in two-level on-chip caching. *Proceedings of the 21st International Symposium on Computer Architecture*, pages 34–45, April 1994.

[12] S. Jourdan, P. Sainrat, and D. Litaize. Exploring configurations of functional units in an out-of-order superscalar processor. *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 117–125, May 1995.

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

[14] Motorola, Inc. MCM36804 8M×36 bit dynamic random access memory module product data sheet. 1995.

[15] Motorola, Inc. MCM6706CR 32k×8 bit static random access memory product data sheet. 1995.

[16] Motorola, Inc. MCM6709BR 64k×4 bit static random access memory product data sheet. 1995.

[17] Motorola, Inc. MCM6726C 128k×8 bit static random access memory product data sheet. 1995.

[18] K. Olukotun, T. Mudge, and R. Brown. Performance optimization for pipelined primary caches. *Proceedings of the 19th International Symposium on Computer Architecture*, pages 181–190, May 1992.

[19] O.A. Olukotun et al. Multilevel optimization in the design of a high-performance GaAs microcomputer. *IEEE Journal of Solid State Circuits*, 26(5):763–767, May 1991.

[20] A. Poursepanj. The PowerPC performance modeling methodology. *Communications of the ACM*, 37(6):47–55, June 1994.

[21] M. Simone et al. Implementation trade-offs in using a restricted data flow architecture in a high performance RISC microprocessor. *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 151–162, June 1995.

[22] T.J. Stanley and T. Mudge. A parallel genetic algorithm for multiobjective microprocessor design. *Proceedings of the 6th International Conference on Genetic Algorithms*, July 1995.

[23] T.J. Stanley and T. Mudge. Systematic objective-driven computer technology optimization. *Proceedings of the 16th Conference on Advanced Research in VLSI*, pages 286–300, March 1995.

[24] R. Uhlig et al. Instruction fetching: Coping with code bloat. *22nd International Symposium on Computer Architecture*, pages 345–356, May 1995.

[25] M. Upton et al. Resource allocation in a high clock rate microprocessor. *Proceedings of ASPLOS-VI*, pages 98–109, October 1994.

[26] M.K. Vernon, E.D. Lazowska, and J. Zahorjan. An accurate and efficient performance analysis technique for multiprocessor snooping cache consistency protocols. *Proceedings of the 15th International Symposium on Computer Architecture*, pages 308–315, June 1988.

[27] B.P. Wilson et al. SUIF: A parallelizing and optimizing research compiler. *ACM SIGPLAN Notices*, 29(12):31–37, December 1994.

[28] K.M. Wilson, K. Olukotun, and M. Rosenblum. Increasing cache port efficiency for dynamic superscalar microprocessors. *Proceeding of the 23rd International Symposium on Computer Architecture*, pages 147–157, May 1996.

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