The Design of a Microsupercomputer

Trevor N. Mudge, Richard B. Brown, William P. Birmingham, Jeffrey A. Dykstra, Ayman I. Kayssi, Ronald J. Lomax, Oyekunle A. Olukotun, and Karem A. Sakallah
University of Michigan

Raymond A. Milano, Vitesse Semiconductor Corporation

As supercomputer performance continues to grow, packaging techniques will remain critical for reducing chip-to-chip delays. In addition, higher integration levels will become increasingly important because they can drastically reduce the number of chip crossings. Microcomputer systems have enjoyed a performance increase of 100 to 200 percent every three years, in large part due to the growth in chip integration density. In contrast, mainframe supercomputers have improved by only about 50 percent every three years.[1] It should not be surprising, then, if the next generation of supercomputers evolves from the microprocessor rather than continuing the mainframe tradition.

This article describes our work to develop a prototype "microsupercomputer" that will realize the best of both the supercomputer and the microprocessor traditions. It does so by using GaAs Mesfet E/D DCFL (gallium arsenide, metal semiconductor field-effect transistor, enhancement/depletion direct-coupled FET logic), a high-speed technology that has good integration density, and by using state-of-the-art packaging technology to prevent chip crossings from dominating the overall speed of the system.

With the levels of integration now becoming available in this technology, it is possible to implement a 32-bit CPU and a floating-point accelerator (FPA) on a single processor chip.[2] It is also possible to implement a 3-nanosecond, 32-Kbit static RAM. These are critical integration levels because they allow the number of chip crossings on the critical timing path of an instruction to be limited to just two (accessing an off-chip cache).

Using advanced GaAs technology and a multichip module package, this prototype next-generation machine takes advantage of the best of both the microprocessor and supercomputer traditions.

The processor, cache, memory controller, and bus interface will be packaged on a single multichip module (MCM) a few inches on a side, to minimize the time penalty of unavoidable chip crossings. High integration levels and high-performance packaging allow the switching speed of GaAs to be reflected in system speed.

The focus of this research is the relationship between hardware implementations and emerging technologies. We chose to implement the MIPS Computer Systems instruction set[3] to bound the architectural options and to eliminate the need to develop compilers and operating systems. In addition, it allows us to use the MIPS MC6280 chassis rather than developing our own high-performance backplane, primary and secondary memory, I/O, and power supplies.

Our efforts are concentrated on developing the processor and cache. The resulting system will be a general-purpose computer that runs a conventional Unix environment and supports standard programming languages and networking protocols. This machine will significantly accelerate execution of the existing large base of application software.

Our circuit simulations of the processor, verified by fabrication and testing of the register file and arithmetic logic unit,[4] indicate that it can be clocked at 250 MHz.

0018-9162/91/0100-0057$01.00 © 1991 IEEE

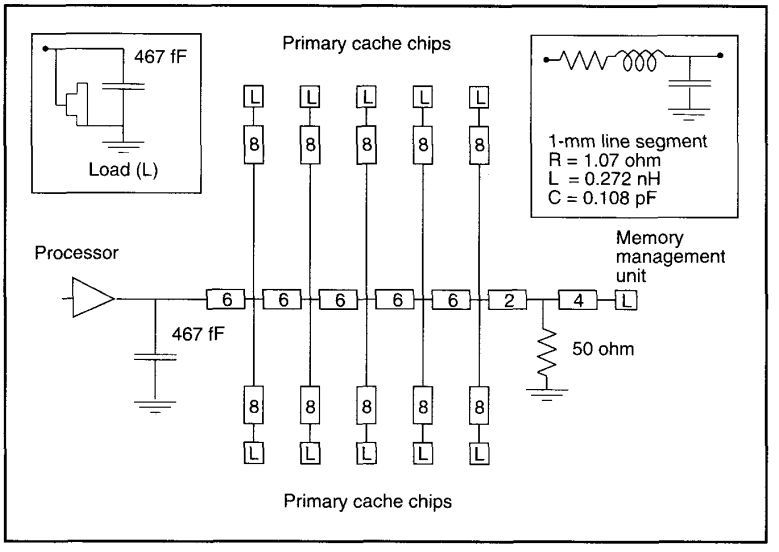


Figure 1. VSPICE schematic to determine CPU-to-cache delay. The number of 1-mm resistor-inductor-capacitor (RLC) segments (inset, upper right) used to model each section of the interconnect is indicated on the lines.

Cache simulations show that a sustained performance of 170 million instructions per second can be expected at this clock rate.

Technology

Technology includes both semiconductors and packaging. To exploit the speed of fast transistors, package parasitics and interconnect lengths must be carefully controlled. Packaging is especially important to Mesfet logic families, since their drive capabilities are inferior to corresponding bipolar families. Though the critical timing path (instruction fetch from first-level cache) has only two chip crossings in our design, the delay associated with them must be on the order of one nanosecond to allow use of a four-nanosecond clock. The need to use high-performance packaging techniques for very fast systems is becoming widely accepted. The choice of semiconductor technology, on the other hand, remains a subject of debate.

Circuit technology. Because of its good speed (130-picosecond gate delays), simplicity (few devices per gate), and low power, we selected E/D DCFL, a Mesfet technology developed by Vitesse Corporation, as the basic logic family for this project. E/D DCFL has circuit topologies similar to E/D NMOS. There are challenges in designing with DCFL:

- limited fan-in and fan-out,
- lack of dynamic circuits due to the Mesfet Schottky gate,
- comparative difficulty of pass transistor design,
- orientation-dependent transistor characteristics, and
- small noise margins.

Nevertheless, dramatic improvement in yield has been achieved; a 30K gate array is now commercially available. This implies that VLSI integration levels are now practical in GaAs DCFL, particularly in a custom design style.

Packaging technology. A major factor in increasing the clock speed is to minimize the round-trip delay to the first-level cache chips, simultaneously maintaining good signal integrity by using a high-performance MCM interconnect technology.[5] The signal lines on the MCM behave as stripline transmission lines and are designed for impedance levels of 50 to 70 ohms. Off-chip drivers must be capable of driving these low impedances.

The multiple fan-out required for the cache prevents ideal impedance matching, making simulation of the signal behavior essential. On the basis of a preliminary layout of the MCM, we performed a VSPICE simulation to estimate interconnect delays. (VSPICE is a proprietary version of SPICE modified by Vitesse Corporation for its E/D Mesfet technology.) We did this by modeling a 1-bit line of a bus by the ladder network shown in Figure 1. We modeled each input pad of the GaAs SRAM chips by a diode-connected Mesfet and parallel capacitor.

In some simulations, the driver was only a pull-up; the terminating resistor acted as a pull-down. In other cases, an active pull-down was used. Round-trip delay times (including the input and output buffers and pads) ranged between 1.6 and 1.8 nanoseconds. Time averaging, with level-sensitive latches, makes it possible to achieve a 4-nanosecond cycle time even though the memory access time is 3 nanoseconds.[6] A three-dimensional transmission-line matrix method,[7] which automatically includes all capacitive and inductive couplings, will be used to characterize more detailed effects of propagation and crosstalk.

Thermal studies have shown that flipchips 1 centimeter square have thermal resistances of around 4- to 5-degrees C/W.[8] The overall dissipation of the 3-inch by 3-inch MCM is on the order of 100W, but most chips on our MCM will dissipate only 4 to 5W. Removing this heat, while not trivial, is within the 1- to 2-W/cm^2 capability of MCM packages.

System architecture

Processor. Central to the design of the microsupercomputer is the 32-bit CPU, which is integrated with the FPA. It is implemented as a five-stage pipeline, with the successive stages denoted IF, RF, ALU, MEM, and WB. Instructions are fetched from cache in the IF stage. One instruction is required every clock period unless there is a cache miss or exception. In the RF stage, the instruction is decoded and operands are read from the register file. In the ALU stage, 32-bit two's complement arithmetic and logic operations take place, and in the MEM stage the data cache is read or written. Finally, in the WB stage, results are written back to the register file. Every operand from memory (except immediates) must be temporarily stored in the register file before it can be used in a calculation. The ALU, IF, and MEM stages are 4 nanoseconds long but, to accommodate the one instruction branch delay expected by the MIPS compiler, the RF and WB stages were made only 2 nanoseconds long.

The basic structure of the CPU is shown in Figure 2. The caches, shown as shaded blocks, are implemented as separate chips. The data path includes a register file with

thirty-one 32-bit general-purpose registers, an ALU, a shifter, and a multiply/divide unit. Note that a separate address adder and incrementer are required for the program counter section to comply with the instruction-per-clock-cycle philosophy. Also shown in Figure 2 are two possible bypass paths, required to avoid hardware interlocks. Bypassing allows the ALU to take operands from any pipe stage, even before the register file is updated with the new value. Nominal delays of major CPU logic blocks are given in Table 1.

Register file. The register file is organized as thirty-one 32-bit registers that can be accessed through three 32-bit ports (one write and two reads). Simultaneous write and read operations to a given register are allowed in the same clock cycle.

ALU. The ALU design is based on a binary look-ahead tree for carry computation.[9] The worst-case instruction is "set on less than" (SLT). It uses the full carry chain plus two additional XOR gates, after which the result is transmitted back across the ALU from most significant bit to least significant bit and then to the top of the ALU for possible bypassing. For branch condition calculations, "quick compare" circuitry (simple logic comparison without subtractions) and a multiplexer to select the comparison type are added to the ALU. The output of the comparison circuit controls the program counter (PC) for the instruction following the branch delay slot.

Shifter. The shifter can perform left and right logical shifts and arithmetic right shifts by 0 to 31 bits. Instead of a barrel shifter, as is commonly used in CMOS, we used a five-stage tree shifter, because at 2.7 nanoseconds the tree design had less than half the delay of a barrel shifter.

Load aligner. The load aligner design is similar to the shifter. An entire data cache access (address generation, cache access, and data alignment) must be done in two cycles, so extra power and area were provided to speed up the address adder and load aligner, allowing a maximum time for cache access.

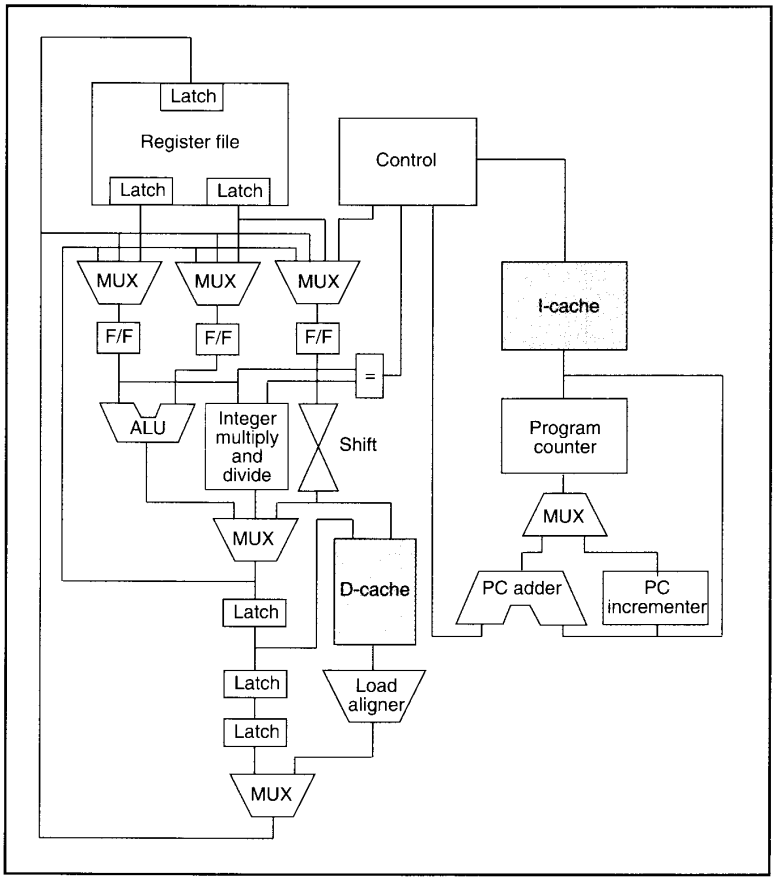


Figure 2. Register-transfer-level of the CPU. Key latches and flip-flops (F/F) are also shown.

Table 1. Parameters of major CPU logic blocks.

Logic Block	Devices	Area	Function	Limiting Delay	Power
Register file	16,085	4.2×1.6 mm	Read + setup	1.6 ns	1.70W
ALU	3,419	3.3×0.5 mm	SLT with bypass	3.5 ns	0.67W
			Quick compare	1.3 ns	
Shifter	1,848	3.4×0.5 mm	Shift data or address	2.7 ns	0.41W
Integer multiply and divide	6,874	4.0×1.0 mm	1 add step	3.6 ns	0.84W
Load aligner	1,922	3.4×0.5 mm	Align	1.1 ns	0.27W
Address adder	1,675	3.2×0.2 mm	Add	1.8 ns	0.21W
PC section	7,082	3.3×1.3 mm	PC + offset	2.5 ns	0.80W
Data path	≈30,000	4.2×6.0 mm			5.00W

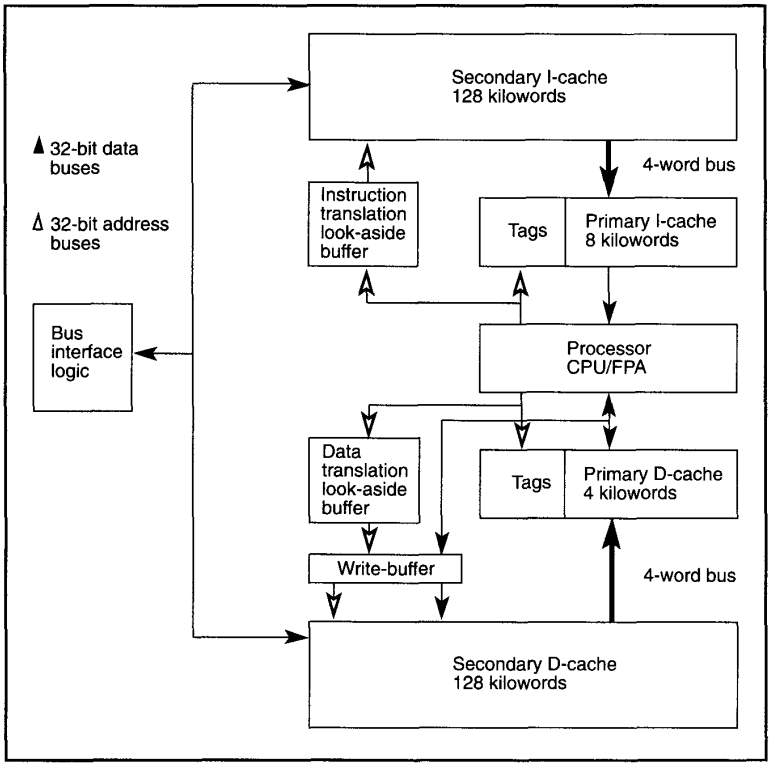


Figure 3. Cache organization.

Controller. The controller decodes instructions and distributes control signals to the register file, ALU, shifter, integer multiply/divide unit, load aligner, and multiplexers. It also contains logic for handling exceptions generated in the CPU, in the FPA, and in the off-chip cache controller. Normally, an exception causes the program counter to jump to the address of the exception-handling routine.[3] Cache miss exceptions are handled by stalling the pipeline. Because of the static design of data path circuits, a stall is produced by simply stopping the clock to the data path.

FPA. The floating-point accelerator will be implemented on the same chip as the CPU. The logic implements single- and double-precision IEEE arithmetic. The principal blocks in the chip are: a register file of thirty-two 32-bit registers, an exponent unit for performing exponent arithmetic and aligning mantissas, an adder/subtracter, a multiplier unit, a divide unit, and a round unit.

Cache architecture. The processor just specified has a 4-nanosecond clock and a potential average CPI (clocks per instruction) of close to one. To realize this potential, it is necessary that the compiler be able to schedule one instruction per clock, and that the memory system be able to deliver one instruction per clock and, when necessary, its operand, too. In other words, we need a high-bandwidth, low-latency memory system. A common way to achieve this in scalar processors is to use a cache and to match the bandwidth requirements set by the peak instruction execution rate of the CPU. This requirement is most easily met with a split, instruction-and-data cache, if pins are not a limitation. Indeed, in systems using conventional CMOS implementations of the MIPS architecture, large split caches (more than 32 kilobytes) are typical. Current CMOS SRAM technology readily supports the construction of caches this size with access times that match the execution rate of current microprocessors.

However, current high-density SRAM technology cannot match a cycle time of 4 nanoseconds. For that, we must look to GaAs SRAMs. Unfortunately, the density levels of these high-speed memories is modest (32 kilobits). Because of signal propagation, access time in a cache is an increasing function of cache size. Keeping access time around 4 nanoseconds limits a cache made of these chips to a size where the performance degradation due to cache misses is unacceptable. The performance can be improved by using a second level of cache.

To validate the preceding rationale for a two-level cache and to determine the optimal design, we developed a simulator, *cacheUM*, described later in this article. Figure 3 shows the organization of the principal components of the memory system that resulted from our experiments with cacheUM. To satisfy the twin requirements of low latency and high bandwidth, we designed the primary cache to deliver both an instruction and a data word within the CPU cycle time (4 nanoseconds) and to be capable of sustaining this in every cycle except for infrequent primary-cache misses.

To simplify concurrent access of instructions and data, we split the primary cache into an instruction cache (I-cache) and a data cache (D-cache). To simplify the organization of the cache, we continued this split in the secondary cache. Apart from the decision to employ a two-level cache, necessitated by the technological limitations mentioned earlier, the guiding philosophy in our memory system design was simplicity, even at the cost of primary-cache miss cycles. The resulting ease of layout and simplicity of control logic more than compensated for these lost cycles by not requiring an extended system cycle time. Key components of the memory organization are summarized in the following paragraphs.

Primary cache. The signal lines from the CPU to the primary caches are critical paths in the design in the sense that their delay determines the lower bound on instruction and data latency. To keep this critical delay to a minimum and avoid the need for a translation look-aside buffer in the access path, both the I-cache and the D-cache are direct-mapped with virtual indices and tags (see Figure 3). The virtual-to-physical address translation is postponed until secondary-cache access, which occurs less frequently. The translation is performed by two logically separate TLBs, the I-TLB on the instruction side and the D-TLB on the data side. They are implemented as 16- and 32-entry direct-mapped SRAM caches, respectively.

The limitations on the size of the primary caches have two sources: signal propagation delays on the MCM between the CPU and the cache chips, and the requirement

that the MIPS architecture support synonyms — two virtual addresses that map to the same physical address. The propagation delays increase as the cache footprint on the MCM increases — that is, as the number of chips in the cache increases. Using the techniques described earlier to determine signal delay on the MCM, we were able to show that our goal of a 4-nanosecond limit on latency places an 8-kiloword (32-kilobyte) limit on the cache. In fact, in a cache of this size the signal delay is close to 50 percent of the cache's latency.

The requirement to support synonyms places a more restrictive limit of 4 kilowords (16 kilobytes) on cache size. However, this limit applies only to the D-cache because the contents of the I-cache cannot be modified by store operations.

The address translation process translates only the top 18 bits of the 32-bit virtual address; consequently, there is no possibility of synonyms occurring within a 16-kilobyte page (defined by the low 14 bits of the virtual address). Therefore, we can avoid the synonyms in the D-cache in a straightforward fashion by using a direct-mapped cache and limiting its size to a 4-kiloword page. The disadvantage is that there is an increase in the miss rate as a result of using a smaller cache.

Methods for allowing larger caches that can contain synonyms require significant additional hardware, such as reverse translation buffers,[10] which would complicate the cache design and layout, resulting in an increase in critical signal delays. These, in turn, would increase cache latency, thus offsetting any advantage gained through improved hit rates.

In our simulations, we found that the best performance was obtained with a cache line of four words. Specifically, just 1.8 percent of the cycles of a typical application program was wasted due to I-cache misses, and, in spite of the 4-kiloword restriction on the D-cache, only 6.88 percent of the cycles was wasted due to D-cache misses (see Table 2).

In keeping with implementation simplicity, stores to the D-cache write through to the secondary cache via a write buffer (see Figure 3) and are assumed to hit the D-cache. If they are found to have missed the primary D-cache, an extra cycle is used to invalidate the entry. This can result in the invalidation of some useful lines in the cache, but our simulations show that the total loss due to D-cache write misses is no more than 0.84 percent of the cycles.

The primary caches are constructed from custom 1K×32-bit SRAM chips, fabricated by integrating four of Vitesse's 1K×8-bit SRAM macrocells on a single die. The 1K×32-bit organization is ideal for the primary cache design. The 8-kiloword I-cache can be realized using eight cache chips for the 2K lines and a further two cache chips for the tag memory for those lines. The 4-kiloword D-cache requires five custom cache chips.

Secondary cache. The secondary cache is constructed from a 12-nanosecond bipolar CMOS SRAM and is also split into instruction and data parts. Each half is 128 kilowords, resulting in a total of 1 megabyte of secondary cache. The split secondary cache is unusual, but it simplifies the logic needed to control the cache. In addition, simulations show that splitting the secondary cache gives a slight (two to five percent) improvement in the CPI figure. The secondary I-cache has a one-line (four-word) path to the primary I-cache. The secondary D-cache is similarly connected to the primary D-cache. These line-wide paths simplify refill when there is a primary-cache miss. Refill takes six cycles.

The secondary caches are connected to the RC6280 backplane through bus interface logic that implements a copy-back and miss-allocate protocol. Secondary-cache misses are expensive: 141 cycles for a clean miss and 235 for a dirty miss. Fortunately, neither is very frequent, wasting only 7.63 percent of total cycles on average.

MCM layout. Figure 4 shows the preliminary layout of the processor and primary cache, the time-critical portion of the MCM. The figure is drawn approximately to scale, with the horizontal dimension representing 5 centimeters. The primary chips are the processor, the cache controller, and the cache chips. The chip placement is designed to minimize the maximum round-trip delay from the CPU to the primary cache. A VSPICE simulation gave the delay time as 1.6 to 1.8 nanoseconds on the worst-case delay path (see the section on packaging technology).

The MCM mounts on the complete processor board that contains the secondary cache and the interface to the RC6280 backplane. With the exception of the processor board, the remainder of the RC6280 implementation is unchanged.

CAD tools

The role of CAD tools in the design of the microsupercomputer was driven by the following goals:

- An integrated approach to the design process, aimed at achieving optimal system performance rather than maximizing the performance of individual subsystems (such as the CPU or the memory subsystems) in isolation. This approach requires the simultaneous consideration of technological, architectural, and packaging issues in one consistent framework.
- The use of existing CAD tools whenever possible, allowing us to leverage ma-

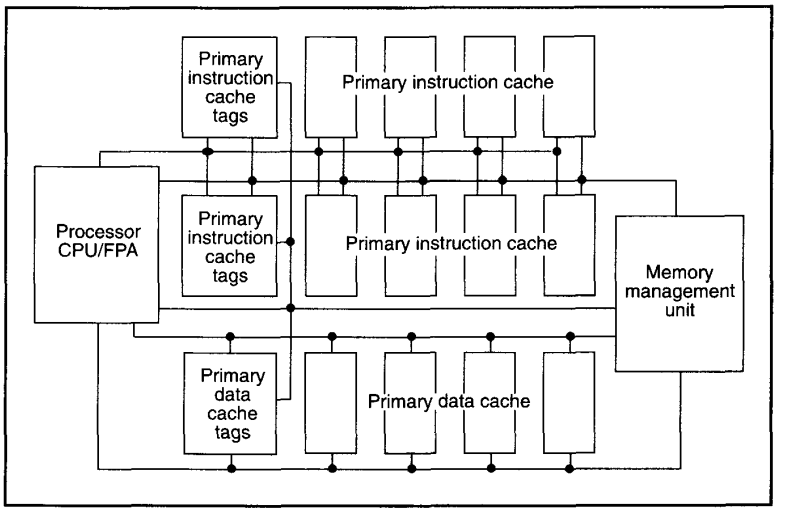


Figure 4. Multichip module layout.

ture CAD technologies such as schematic capture, logic and circuit simulation, and automatic layout systems. We use Mentor Graphics' schematic capture and simulation tools. For circuit simulation, we use VSPICE, the Vitesse version of SPICE mentioned earlier. For high-level simulation, we use Cadence Design Systems' Verilog simulator. And, for layout, we are using Seattle Silicon Corporation's Chip Crafter and Finesse tools, customized with generators for the four-metal Vitesse GaAs process.

• Development of new CAD tools to address specific needs in our design for which current tools are either nonexistent or inadequate. These tools include a finite-difference transmission-line-matrix simulator to characterize the chip-to-chip interconnect on the MCM[7]; *checkTc* and *minTc*, two timing analysis and optimization tools to help identify critical paths in the design and to minimize the cycle time[6]; and a cache simulator to study various architectural trade-offs.

In this section, we highlight the cache simulation tool, cacheUM. The efficiency with which a CPU architecture executes a program can be measured by the average number of CPI.[1] This value is calculated by dividing the number of useful instructions in a program by the number of cycles it takes for the CPU architecture to execute the program. For a particular clock speed, CPI also gives a measure of total processor performance.

As noted earlier, a high-performance processor requires a matching high-performance memory system. In order to make intelligent memory system design choices, we must evaluate the CPI values for different memory organizations. The tool developed to make these evaluations is cacheUM, a trace-driven two-level cache simulator that models all aspects of the memory system described earlier.

Simulation with cacheUM operates as follows: MIPS object code is annotated by MIPS Pixie[11] at basic-block entry points and at memory references. When this annotated code is executed, it produces a trace of instruction and data addresses. This trace stream is fed to the cache simulator, which generates and collects statistics. The cycle count that cacheUM provides assumes that each instruction executes in the CPU in one cycle. This cycle count must be added to the number of internal CPU stall cycles caused by executing multicycle instructions such as integer multiply, integer divide, and floating-point instructions. The number of extra cycles is provided by MIPS Pixstats. This augmented cycle count is used to calculate the value of CPI and the contribution of the different parts of the system to the value of CPI for the memory organization. The cache simulator executes an average of 80 times slower than the original object file.

The applications used to provide the traces have a significant effect on the performance of a memory system. We have used a mix of integer and floating-point applications to represent the work load of a high-performance workstation in a technical environment. We execute each application assuming a context switch interval of one million instructions. At each context switch interval, all caches are flushed. This technique of simulation context switches is shown to give pessimistic performance results.[12] We calculate the composite CPI value by dividing the total number of cycles by the number of instructions executed. This provides the harmonic mean of the CPI weighted by the number of instructions executed by each benchmark.

Table 2 shows the statistics obtained by executing 15 common benchmarks. These statistics include the number of dynamic instructions executed (in millions), the number of cycles it took to execute them (in millions), and the top four memory system contributors to performance degradation (shown as a percentage of CPI). From this table, we note that primary D-cache misses (PD-miss) account for most of the loss in memory system performance, a direct consequence of restricting the D-cache size to a page, as explained earlier. However, the performance loss due to secondary D-cache misses (SD-miss) is not much less than that due to primary D-cache misses. We believe that the secondary cache miss rates are artificially high due to the manner in which context switches were simulated. In reality, it is possible for several processes, depending on the size of their

Table 2. Benchmark performance results.

Name	Instructions*	Cycles*	PI-miss**	PD-miss**	SI-miss**	SD-miss**	CPI
5Diff	218.3	306.6	0.06	9.88	0.31	12.15	1.40
Awk	34.9	50.6	4.90	3.55	2.27	2.43	1.45
Doducd	48.1	102.6	9.14	8.57	8.95	3.41	2.13
Dhrystone02	53.6	64.2	0.07	0.02	0.30	0.14	1.20
Espresso	119.0	149.8	1.92	3.58	1.88	2.59	1.26
Gnuchess	488.2	606.6	0.86	0.84	2.99	1.49	1.24
Grep	49.4	59.1	0.05	0.14	0.29	0.30	1.20
Linpackd	4.0	7.1	0.17	24.69	0.86	8.11	1.78
LFK12	275.5	394.5	0.01	2.52	0.06	0.91	1.43
Nroff	15.7	21.9	0.66	1.21	2.59	2.07	1.40
Small	16.7	22.4	0.08	2.45	0.36	1.37	1.34
SPICE2g6	297.3	552.9	5.58	19.31	3.35	14.03	1.86
Whetd	9.4	18.0	0.27	0.10	1.28	0.48	1.91
Wolf33	83.2	180.2	3.38	20.98	5.06	11.70	2.16
YACC	96.9	136.0	0.31	5.89	0.67	4.68	1.40
Total	1,810.4	2,672.6	1.80	6.88	2.13	5.47	1.48

* In millions.

** Percentage of CPI.

working sets, to share a 1-megabyte secondary cache without flushing each other's entries entirely between context switches.

The performance goals of the prototype 250-MHz microsupercomputer require an integrated design approach in which technology, architecture, and packaging are considered simultaneously. GaAs DCFL technology, with its high speed, high level of integration, and high yield, is an important element in achieving the desired performance.

Multichip-module packaging must be used to achieve the needed cache performance, and careful partitioning of the processor components is required to minimize the number of chip crossings in the critical path.

The use of CAD tools is critical. It is important to use an automatic layout system to leverage the technology and take immediate advantage of continuously increasing integration levels without extensive and costly redesign.

Simulation of cache performance is necessary to achieve the best compromise between size and speed and, in general, the use of simulation is crucial in making the majority of decisions along the design path.

Coordinating these different aspects makes it possible to achieve a global optimization of the design and to build a system meeting the specifications we have described.■

Acknowledgments

We thank MIPS Computer Systems for supporting this project with technical assistance under a special licensing arrangement. We gratefully acknowledge the assistance of Seattle Silicon Corporation and Mentor Graphics Corporation. This work has been supported in part by the Defense Advanced Research Projects Agency under DARPA/ARO contract DAAL03-90-C-0028, and by the US Army Research Office under the URI Program contract DAAL03-87-K-0007.

References

1. J.L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, Calif., 1990.
2. H. Vlahos and V. Milutinovic, "GaAs Microprocessors and Digital Systems," *IEEE Micro*, Vol. 8, No. 1, 1988, pp. 28-56.
3. G. Kane, *MIPS RISC Architecture*, Prentice Hall, Englewood Cliffs, N.J., 1988.
4. J.A. Dykstra, *High-Speed Microprocessor Design with Gallium Arsenide Very Large Scale Integrated Digital Circuits*, PhD dissertation, Univ. of Michigan, 1990.
5. H.B. Bakoglu, *Circuits, Interconnects, and Packaging for VLSI*, Addison-Wesley, Reading, Mass., 1990.
6. K.A. Sakallah, T.N. Mudge, and O.A. Olukotun, "*checkTc* and *minTc*: Timing Verification and Optimal Clocking of Synchronous Digital Circuits," *Proc. ICCAD 90, Int'l Conf. on Computer-Aided Design*, IEEE Computer Soc. Press, Los Alamitos, Calif., Order No. 2055, pp. 552-555.
7. R.H. Voelker and R.J. Lomax, "A Finite-Difference Transmission Line Matrix Method Incorporating a Nonlinear Device Model," *IEEE Trans. Microwave Theory and Techniques*, Vol. 38, No. 3, Mar. 1990, pp. 302-312.
8. Y.C. Lee et al., "Internal Thermal Resistance of a Multi-Chip Packaging Design for VLSI Based Systems," *IEEE Trans. Components, Hybrids, and Manufacturing Technology*, Vol. 12, No. 2, June 1989, pp. 163-169.
9. N.H.E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, Reading, Mass., 1985.
10. J.R. Goodman, "Coherency for Multiprocessor Virtual Address Caches," *Proc. ASPLOS, Second Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, Oct. 1987, pp. 72-81.
11. *RISCompiler Languages Programmer's Guide*, MIPS Computer Systems, Inc., Santa Clara, Calif., Dec. 1988.
12. A. Agarwal, *Analysis of Cache Performance for Operating Systems and Multiprogramming*, Kluwer Academic Publishers, Boston, Mass., 1988.

Trevor N. Mudge is an associate professor of electrical engineering and computer science at the University of Michigan, which he joined in 1977, and is the director of the university's Advanced Computer Architecture Laboratory. He has written more than 100 papers on computer architecture, programming languages, VLSI design, and computer vision.

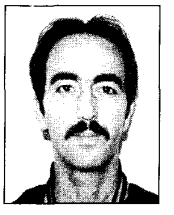
Mudge received the BSc in cybernetics from the University of Reading, England, in 1969, and the MS and PhD in computer science from the University of Illinois in 1973 and 1977, respectively. He is a senior member of the IEEE, and a member of the IEEE Computer Society, the ACM, and the British Computer Society.

Richard B. Brown joined the faculty of the University of Michigan Department of Electrical Engineering and Computer Science in 1985 and has been involved in shaping its VLSI program and introducing a uniform set of CAD tools into the curriculum.

Brown received the BS and MS degrees in electrical engineering from Brigham Young University in 1976 and the PhD in electrical engineering from the University of Utah in 1985. He is a member of the IEEE and the IEEE Computer Society.

William P. Birmingham is an assistant professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, where he is also a member of the Advanced Computer Architecture Laboratory and the Artificial Intelligence Laboratory. He has spent several years in industry, developing AI-based CAD tools.

Birmingham received the BS in electrical engineering in 1982, the MS in computer engineering in 1983, and the PhD in electrical engineering in 1988, all from Carnegie Mellon University. He is a member of the IEEE, the IEEE Computer Society, the ACM, and Sigma Xi.



Jeffrey A. Dykstra has worked on advanced IC design of communication devices at Motorola's Chicago Corporate R&D division since June 1990.

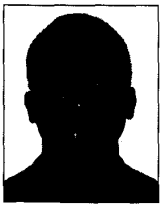
Dykstra received the BS in letters and engineering from Calvin College in 1981. He received the BSEE in 1982 and the MSEE in 1984, both from the University of Michigan, where he completed the requirements for the PhD in October 1990. He is a member of Eta Kappa Nu, Tau Beta Pi, and the IEEE.

Ayman I. Kayssi is a member of the Advanced Computer Architecture Laboratory and a PhD candidate in the Department of Electrical Engineering and Computer Science at the University of Michigan. His research interest is computer-aided design for high-speed VLSI circuits and systems.

Kayssi received the BE degree in 1987 from the American University of Beirut, Lebanon, and the MSE degree in 1989 from the University of Michigan. He is a student member of the IEEE and the IEEE Computer Society.

Ronald J. Lomax is a professor of electrical engineering and computer science at the University of Michigan, where he has worked since 1961. He conducts research on high-speed GaAs digital circuits in the Center for High-Frequency Microelectronics. He has published works on electron gun design, plasma simulation, microwave devices, finite-element simulation of solid-state devices, and CAD for VLSI.

Lomax received the BA in mathematics in 1956 and the MA and PhD in applied mathematics in 1960, all from the University of Cambridge, England. He is a fellow of the Cambridge Philosophical Society, a senior member of the IEEE, and a member of the IEEE Computer Society and the Society for Industrial and Applied Mathematics.



Oyekunle A. Olukotun is pursuing a PhD at the University of Michigan. His research interests include parallel algorithms for computer-aided design of integrated circuits and tools for the analysis, design, and verification of high-performance digital systems.

Olukotun received the BS in electrical engineering in 1985 and the MS in computer engineering in 1987, both from the University of Michigan. He is a student member of the IEEE Computer Society.

Karem A. Sakallah has been an associate professor of electrical engineering and computer science at the University of Michigan since 1988. His research focuses on CAD for integrated circuits and systems.

Sakallah received the BE in electrical engineering from the American University of Beirut in 1975 and the MSEE and PhD in electrical and computer engineering from Carnegie Mellon University in 1977 and 1981, respectively. He is a member of the IEEE Computer Society.

Raymond A. Milano is the director of standard cells and foundry services for custom and semicustom GaAs products at Vitesse Semiconductor Corporation, which he joined in 1984.

Milano received the BS from the University of Rhode Island in 1972, the MS from Rutgers University in 1974, and the PhD from the University of Illinois in 1980, all in electrical engineering.

Readers may write to all the authors (except Milano) at the Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, MI 48109-2122. Milano can be reached at Vitesse Semiconductor Corporation, 741 Calle Plano, Camarillo, CA 93010.