

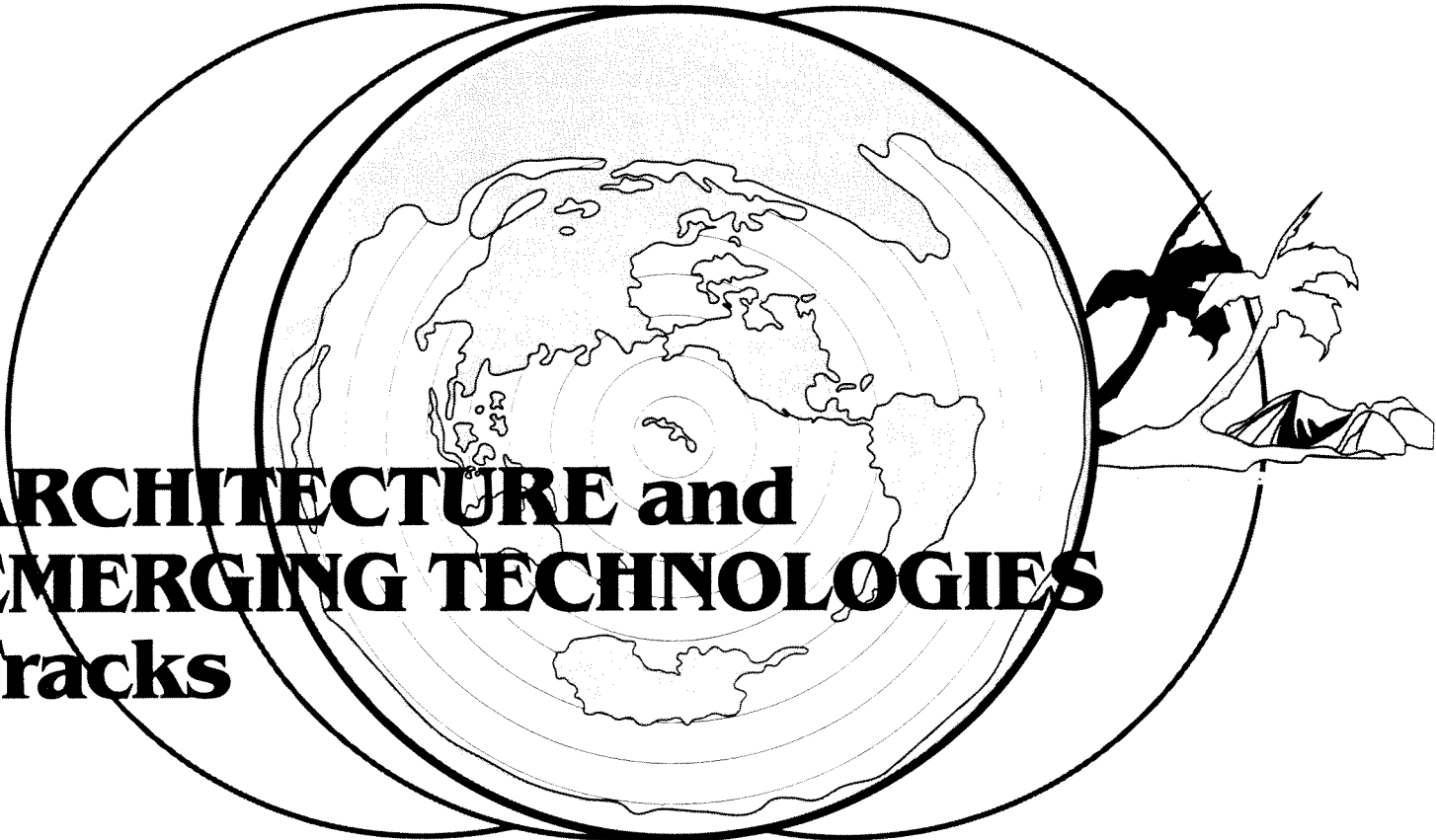
Proceedings of the Twenty-Fourth Annual Hawaii International Conference on

System Sciences

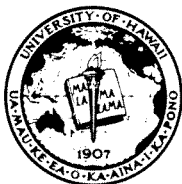
Edited by
VELJKO MILUTINOVIC
BRUCE D. SHRIVER

1991

Vol I



ARCHITECTURE and EMERGING TECHNOLOGIES Tracks



IEEE Computer Society Press



The Institute of Electrical and Electronics Engineers, Inc.

The Design of a GaAs Micro-Supercomputer

T. N. Mudge, R. B. Brown,
W. P. Birmingham, J. A. Dykstra, A. I. Kayssi, R. J. Lomax,
O. A. Olukotun, and K. A. Sakallah
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122

R. Milano
Vitesse Semiconductor Corp.
741 Calle Plano
Camarillo, CA 93010

Abstract

This paper is an overview of the architecture, technology and CAD tools used in the design of an experimental 250 MHz "micro-supercomputer" which is being designed for a sustained performance of 170 MIPS. The system will include a gallium arsenide processor which executes the MIPS instruction set and a two-level cache memory system, packaged on a multi-chip module. The risk in undertaking this project is minimized by using existing but advanced GaAs technology, by building needed CAD tools on top of commercial tools, and by using a standard instruction-set architecture.

1 Introduction

As supercomputers continue to grow in performance, 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. The growth in integration density has been a major factor in the performance increase enjoyed by microprocessor systems, of 2-3 times every three years. In contrast, mainframe supercomputers have only improved by about 1.5 times every three years [1]. It should not be surprising, then, if the next generation of supercomputers evolves from the microprocessor rather than the mainframe supercomputer tradition.

This paper describes work to develop a prototype "micro-supercomputer" that will realize the best of both the supercomputer and the microprocessor traditions by us-

ing GaAs MESFET enhancement/depletion direct-coupled FET logic (E/D DCFL), 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 level 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. It is also possible to implement a 3 ns 32 K-bit SRAM. 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). The processor, cache, memory controller, and bus interface will be packaged on a single multi-chip module (MCM) of 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 [2] instruction set to bound the architectural options, and to eliminate the need for developing 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 UNIX¹ environment and supports standard programming languages and networking

¹UNIX is a trademark of AT&T.

protocols. A large base of application software exists, the execution of which could be accelerated significantly on this machine.

In Section 2 of this paper, circuit and packaging technology are discussed. Our SPICE simulations of the CPU, verified by fabrication and testing of the register file and arithmetic logic unit [3], indicate that it can be clocked at 250 MHz. For a system to make effective use of a processor running at this speed, the memory must be capable of delivering both instructions and data, with low latency, at a rate of 250 MHz. Section 3 outlines a systems architecture that takes advantage of the GaAs technology, and discusses trade-offs in the cache memory design. Section 4 describes the CAD tools developed for this project. In Section 5 we summarize the challenges in the design of a micro-supercomputer, and conclude that solutions to these are within the reach of current technology.

2 Technology

2.1 Circuit 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 these must be on the order of 1 nS to allow use of a 4 nS 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.

Because of its good speed (130 pS gate delays), simplicity (few devices per gate), and low power, we selected a MESFET technology developed by Vitesse, E/D DCFL, as the basic logic family for this project. E/D DCFL has circuit topologies similar to E/D NMOS, although with lower fan-in and fan-out. Figure 1 shows typical DCFL logic gates. Simplicity and low power requirements make possible a high integration level, which is necessary for MESFET logic to achieve system speed commensurate with its gate speed. The chips are designed to run over a temperature range of 0–70° C and static NOR logic is used primarily.

Early problems with GaAs material quality and processing yield have left lingering questions about the practicality of large scale integration. This is particularly true for

DCFL which has small logic swings and small noise margins, because the Schottky barrier gates draw current on the logic high input. This makes DCFL very dependent upon enhancement threshold uniformity to maintain noise margins. Factors which affect yield are wafer quality, processing techniques and operator skill, number of processing steps (mask levels), die size and number of pads, critical dimensions, circuit complexity, and design methodology. The Vitesse E/D DCFL process is designed to produce high yield, as well as low power dissipation, high speed, and a high integration level.

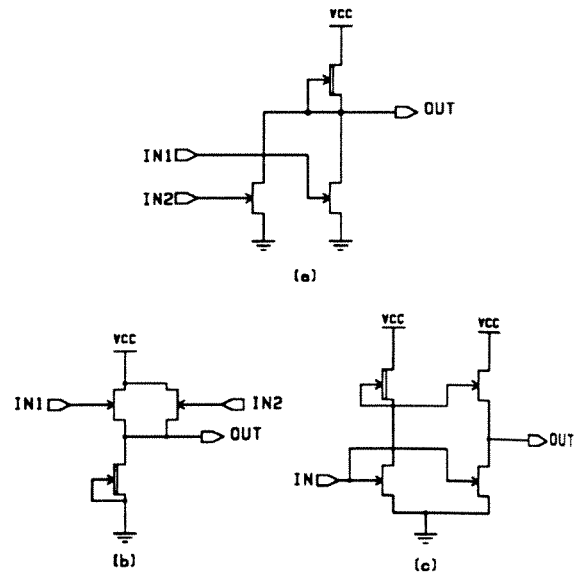


Figure 1: Example DCFL gates. (a) NOR gate, (b) OR gate, (c) Super-buffer.

The use of self-aligned transistors and refractory metal gates minimizes the number of masks required to make high-transconductance devices with adequate threshold voltage control. The four-layer aluminum interconnect system is based on industry standard techniques and makes use of no special equipment or technology. With this process, GaAs ICs having low power dissipation and ECL-like speed can be built using about half the mask layers of current ECL processes.

Figure 2 shows the dramatic improvement in yield which has been achieved as a function of time, with data shown for four gate-arrays: a 1500 gate-array part, a 4500 gate-array part, a 10K gate-array part, a 30K gate-array part. This data demonstrates that VLSI integration levels are practical in GaAs DCFL, and suggests yields and integration levels will continue to improve. There are, however,

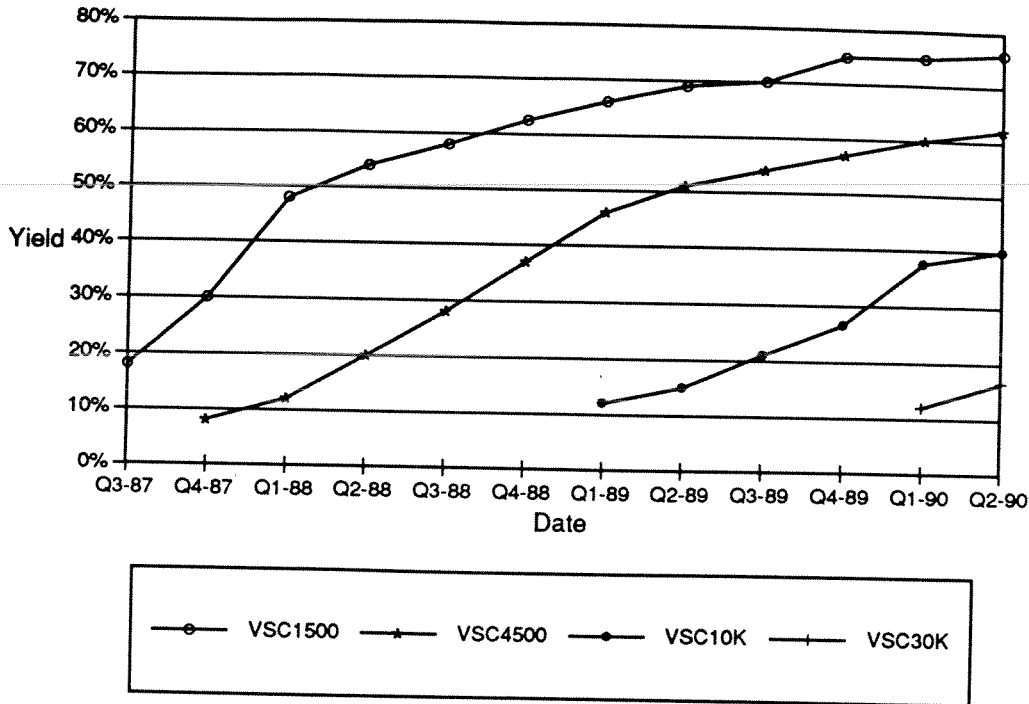


Figure 2: Functional probe yield.

challenges in designing with DCFL compared to NMOS or CMOS: 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.

In summary, even though compared to NMOS or CMOS, DCFL still presents the designer with several limitations that must be worked around, the level of integration becoming available through yield improvements in DCFL GaAs makes implementation of a 32-bit processor on a single chip possible. This is a critical integration level for reducing the number of chip crossings and gives DCFL and edge over ECL.

2.2 Packaging Technology

A major factor in increasing the clock speed is to minimize the roundtrip delay to the first-level cache chips, while simultaneously maintaining good signal integrity by using a high performance MCM interconnect technology [4]. The signal lines on the MCM behave as stripline transmission lines and are designed for impedance levels of 50-70 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, a VSPICE² simulation was performed to estimate interconnect delays. This was done by modeling a one-bit line of a bus by the ladder network shown in Fig. 3. Each input pad of the GaAs SRAM chips was modeled by a diode-connected MESFET and parallel capacitor. In some simulations the driver was only a pullup; the terminating resistor acted as pulldown. In other cases an active pulldown was used. Roundtrip delay times (which include the input and output buffers and pads) ranged between 1.6 and 1.8 ns. Time averaging, with level-sensitive latches, makes it possible to achieve a 4 ns cycle time even though the memory access time is 3 ns [5]. A three-dimensional transmission-line matrix method [6] which automatically includes all capacitive and inductive couplings will be used to optimize further and to characterize more detailed effects of propagation and crosstalk.

The chips must be incorporated into the package in a way which minimizes lead length and maximizes heat transfer from the chip. Possibilities include wire bonding, flip chip and tape automated bonding. Thermal studies on a silicon processor module [7] have shown that flip chips 1 cm square have thermal resistances of around 4-5 °C/W. Be-

²VSPICE is a proprietary version of SPICE modified by Vitesse Corp. for their E/D MESFET technology.

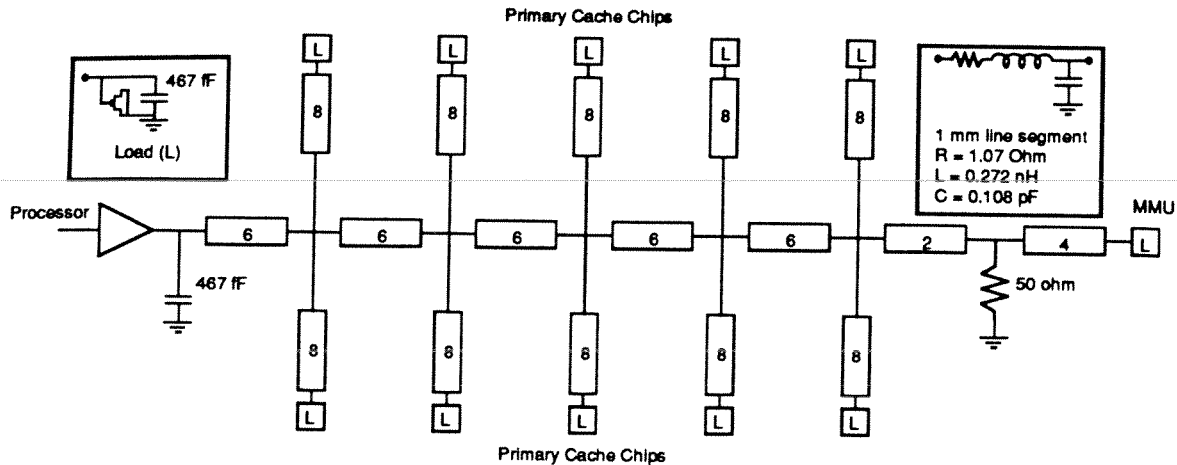


Figure 3: VSPICE schematic to determine CPU-cache delay. The number of 1 mm RLC segments used to model each section of the interconnect is indicated on the lines.

cause most of the heat is generated on the bottom of the flipped chip, GaAs chips are expected to have similar thermal resistances. An approximate estimate for wire bonded chips gives about twice these values, but still within a reasonable range. The overall dissipation of the 3in. x 3 in. MCM is on the order of 100 W, but most chips on our MCM will dissipate only 4-5 W. Removing this heat, while not trivial, is within the 1-2 W/cm² capability of MCM packages.

This suggests that a clear-cut decision cannot be made on a purely thermal basis. The lower parasitics and higher wiring density of flip chips should give superior performance. Fortunately, availability of flip chip packaging is improving rapidly.

3 System Architecture

3.1 Processor

Central to the design of the micro-supercomputer is the 32-bit CPU, which is integrated with the FPA. It is implemented as a five-stage pipeline shown in Fig. 4. The successive stages are denoted IF, RF, ALU, MEM, 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. 32-bit two's complement arithmetic and logic operations take place in the ALU stage, and the data cache is read or written during the MEM stage. Finally, during the WB stage, results are written back to the register file. Every

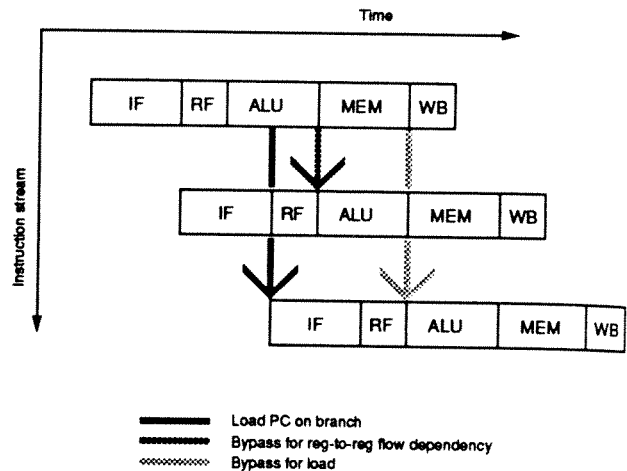


Figure 4: Pipeline stages and relative timing.

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 ns long, but to accommodate the one instruction branch delay expected by the MIPS compiler, the RF and WB stages were made only 2 ns long. Also shown in Fig. 4 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.

The basic structure of the CPU is shown in Fig. 5. The caches, shown as shaded blocks, are implemented as separate chips. The data path includes a register file with thirty-

Logic Block	Devices (Area)	Function	Delay	Power Estimate
Register File SRAM-like 3-port, 32-Registers, 32-bits	16,085 (4.2 x 1.6 mm)	Read Read Set-Up Pre-Set Write (from clk) Write (from data) Write Set-Up	0.4 nS 1.2 nS 1.5 nS 1.2 nS 0.6 nS 0.8 nS	1.70W
Arithmetic Logic Unit Binary carry look-ahead Functions: ADD, SUB, AND, NOR, OR, SLT, BRANCH CONDITION	3,419 (3.3 x 0.5 mm)	Add SLT (Set on <) SLT with bypass Quick Compare	2.5 nS 2.9 nS 3.5 nS 1.3 nS	0.67W
Shifter Functions: SLL, SRL, SRA	1,848 (3.4 x 0.5 mm)	Data Control	2.7 nS 2.5 nS	0.41W
Integer Multiply and Divide Functions: MULT(U), DIV(U)	6,874 (4.0 x 1.0 mm)	1 add step	3.6 nS	0.84W
Load Aligner Functions: LB(U), LH(U), LWL, LWR	1,922 (3.4 x 0.5 mm)	Align	1.1 nS	0.27W
Address Adder Group 4 carry look-ahead	1,675 (3.2 x 0.2 mm)	Add	1.8 nS	0.21W
PC section	7,082 (3.3 x 1.3 mm)	PC + Incr. PC + Offset	2.3 nS 2.5 nS	0.80W
Datapath	$\approx 30,000$ (4.2 x 6 mm)			5W

Table 1: Summary of parameters for major CPU logic blocks.

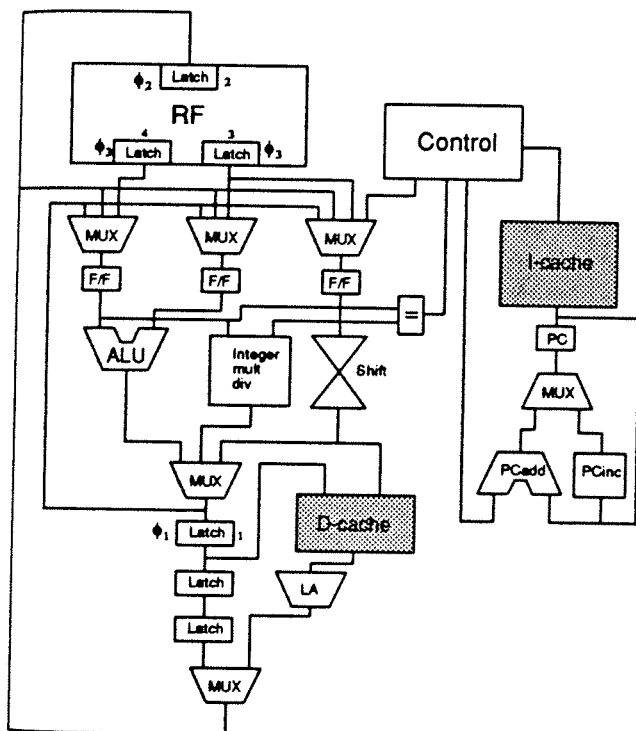


Figure 5: Register-transfer level block diagram of the CPU.

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. Nominal delays of major CPU logic blocks are given in Table 1; brief descriptions of these and other blocks follow.

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 [8]. 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 MSB to LSB, 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 multiplexor to select the comparison type are added to the ALU. The output of the comparison circuit controls the 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 ns, 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 devoted to speed up the address adder and load aligner, allowing a maximum time for cache access.

Controller. The controller decodes instructions and distributes control signals to the register file, ALU, shifter, integer multiply/divide unit, load aligner, and multiplexors. 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 [2]. Cache miss exceptions are handled by stalling the pipeline. Because of the static design of datapath circuits, a stall is produced by simply stopping the clock to the datapath.

FPA. The FPA 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 mantissae, an adder/subtractor, a multiplier unit, a divide unit, and a round unit. To preserve the same latencies as the RC6280 requires about 45,000 gates. Although this is three times the gate count required by the CPU (which was designed in two-year-old version of DCFL) current processing technology can support it (see Fig. 2). The layout for the FPA will be done using the the GaAs compiler described in Sec. 4.1.

3.2 Cache Architecture

The processor specified above has a 4 ns clock and a potential average CPI (clocks per instruction) of close to one. To realize this potential it is necessary that: 1) the compiler can normally schedule one instruction per clock; and 2) the memory system can 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. Except for pin-out limitations, this requirement is most easily met with a

split instruction/data cache. Indeed, in systems built using conventional CMOS implementations of the MIPS architecture, large split caches (32-K+ bytes) are typical. Current CMOS SRAM technology readily supports the construction of caches of this size with access times (t_{ac}) that match the execution rate of current microprocessors. However, current high density SRAM technology cannot match a cycle time of 4 ns. For that we must look to GaAs SRAMs. Unfortunately, the density levels of these high-speed memories is modest (32 K-bits), so the placement of the memory chips becomes critical, as it has a significant impact on cache access time. Thus, the cache access time becomes an increasing function of the cache size or equivalently of the cache hit rate. Clearly, there is a point of diminishing returns. Consider the memory access time given by the expression:

$$ht_{ac} + (1 - h)T$$

where h is the cache hit rate and T is the time penalty for a cache miss. If ht_{ac} is fixed as a consequence of h being a decreasing function of t_{ac} , then the miss penalty becomes a dominant factor in determining the CPI for the system. The miss penalty T can be reduced in a straightforward manner by using a second level of cache. The alternative is to reduce the access time to main memory which is not a cost effective solution.

To validate the above rationale for a 2-level cache the simulator, **cacheUM** described in Section 4.4, was developed to determine the optimal design. Figure 6 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, the primary cache was designed to deliver both an instruction and a data word within the CPU cycle time (i.e., 4 ns) and to be capable of sustaining this every cycle except for infrequent primary cache misses. To simplify concurrent access of instructions and data, the primary cache is split into an instruction cache (I-cache) and a data cache (D-cache). To simplify the organization of the cache this split was continued into the secondary cache. Apart from the decision to employ a 2-level cache, which was necessitated by the technological limitations mentioned above, 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 in the memory organization are summarized in the following paragraphs.

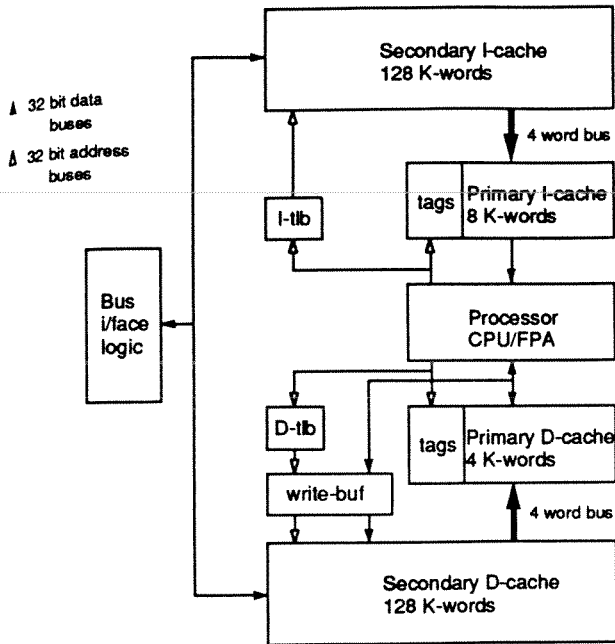


Figure 6: Block diagram of the cache organization.

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, both the I- and D-cache are direct mapped with virtual indices and tags, to avoid the need for a translation lookaside buffer (TLB) in the access path (see Fig. 6). 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 directed mapped SRAM caches.

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 footprint of the cache on the MCM increases, i.e., as the number of chips in the cache increases. Using the techniques described in Section 2.2 to determine signal delay on the MCM, we were able to show that our goal of a 4 nS limit on latency places an 8 K-word (32 K-bytes) limit on the cache. In fact, in a cache of this size the signal delay is close to 50% of the cache's latency.

The requirement to support synonyms places a more re-

strictive limit of 4 K-words (16 K-bytes) on cache size. However, this limit only applies to the D-cache because the contents of I-cache cannot be modified by store operations. The address translation process only translates the top 18 bits of the 32-bit virtual address, consequently there is no possibility of synonyms occurring within a 16 K-byte page (defined by the low 14 bits of the virtual address). Therefore, the synonyms in the D-cache can be avoided in a straightforward fashion by using a direct mapped cache and limiting its size to a 4 K-word 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 [9], which would complicate the cache design and layout, resulting in an increase in critical signal delays. These, in turn, will increase cache latency and so work towards offsetting any advantage gained through improved hit rates. In our simulations we found that the best performance was obtained a cache line of four words. Specifically, just 1.8% of the cycles of a typical application program was wasted due to I-cache misses and, in spite of the 4 K-word restriction on the D-cache, only 6.88% of the cycles were wasted due to D-cache misses (see Section 3.2).

In keeping with implementation simplicity, stores to the D-cache write-through to the secondary cache via a write buffer (see Fig. 6) 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 some useful lines in the cache being invalidated, but our simulations show that the total loss due to D-cache write misses is no more than 0.84% of the cycles.

The primary caches are constructed from custom 1 K × 32-bit SRAM chips. The chips will be fabricated by integrating four of Vitesse's 1 K × 8-bit SRAM macro-cells on a single die. The 1 K × 32-bit organization is ideal for the primary cache design. The 8 K-word I-cache can be realized using 8 cache chips for the 2 K-lines and a further 2 cache chips for the tag memory for those lines. The custom cache chips also include a comparator to facilitate their use in tag memory. The 4 K-word D-cache requires 5 custom cache chips.

Secondary Cache. The secondary cache is constructed from 12 ns BiCMOS SRAM, and is also split into instruction and data parts. Both halves are 128 K-words each, resulting in a total of 1 M-byte 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 (2–5%) improvement in the CPI figure. The secondary I-cache

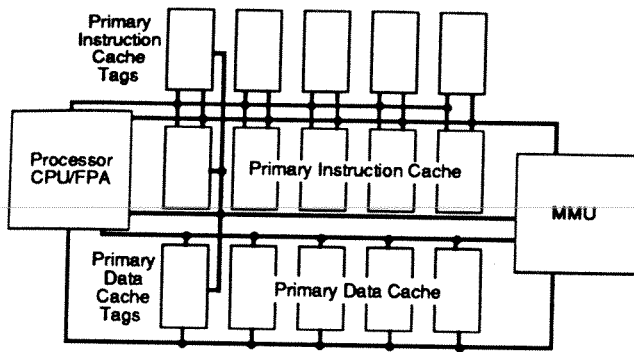


Figure 7: MCM layout.

has a one line (4 word) path to the primary I-cache. The secondary D-cache is connected the primary D-cache in a similar fashion. These line-wide paths simplify refill when there is a primary cache miss. Refill takes 6 cycles.

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

3.3 MCM Layout

Figure 7 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 width representing 5 cm. The primary chips are the processor, the cache-controller, and the cache-chips. The chip placement is designed to minimize the maximum roundtrip delay from the CPU to the primary cache. A VSPICE simulation gave the delay time as 1.6–1.8 ns on the worst case delay path (see Section 2.2).

The MCM mounts on the complete processor board that contains the secondary cache and the interface to the R6000 backplane. With the exception of the processor board power supply, the remainder of the R6000 implementation is unchanged.

4 CAD Tools

The role of CAD tools in the design of the micro-supercomputer was driven by the following goals:

- An integrated approach to the design process aimed at achieving optimal *system* performance as opposed

to maximizing the performance of individual subsystems (e.g. the CPU or the memory subsystems) in isolation. Such an 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 mature 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 the Verilog simulator³. And for layout, we are using the ChipCrafter and Finesse⁴ tools, customized with generators for the 4-metal Vitesse GaAs process.
- Development of new CAD tools to address specific needs in our design for which current tools are either non-existent or inadequate. These tools include a finite-difference transmission-line-matrix simulator to characterize the chip-to-chip interconnect on the MCM [6], *checkT_c* and *minT_c*, two tools for timing analysis and optimization to help identify critical paths in the design and to minimize the cycle time [5], and a cache simulator to study various architectural trade-offs.

4.1 GaAs Compiler

Several projects to design supercomputers have relied on gate-array technology for rapid implementation. However, the fixed location of the gates inevitably leads to sub-optimal integration levels and long interconnections. The compiler concept, however, allows the designer to achieve near-custom level use of chip area without the effort necessary for custom design. In the case of GaAs there is an additional pay-off because of the reduction in inter-gate delay, which represents a much greater percentage of the overall system delay relative to slower technologies. Tools for silicon compilation from Seattle Silicon Corporation which were available included the ChipCrafter IC design tool, which was set up for use with 2-metal CMOS technologies. To make use of this tool in the design of GaAs circuits, we used the Compiler Development System (CDS) that complements ChipCrafter to build a database of GaAs cells, gates, memory elements, high-level modules, and datapath cells. CDS allows its users to write code in *C* to generate the layout, the logic simulation model, and the symbol for

³Verilog is a trademark of Cadence Design Systems.

⁴ChipCrafter and Finesse are trademarks of Seattle Silicon Corp.

the cell being generated. The technology-independent logic synthesizer in ChipCrafter, called Finesse, works with the GaAs library to synthesize circuits with available cells. A technology map file for GaAs specifies available cells and their relative performance/area costs. The original 2-metal router was replaced with a multi-level metal router for the GaAs modules generated for the Vitesse process. Timing data for the GaAs cells is currently being extracted from repeated VSPICE runs in order to simulate the individual cells for different input slew rates and load capacitances, so that it will be compatible with the timing analyzer in ChipCrafter.

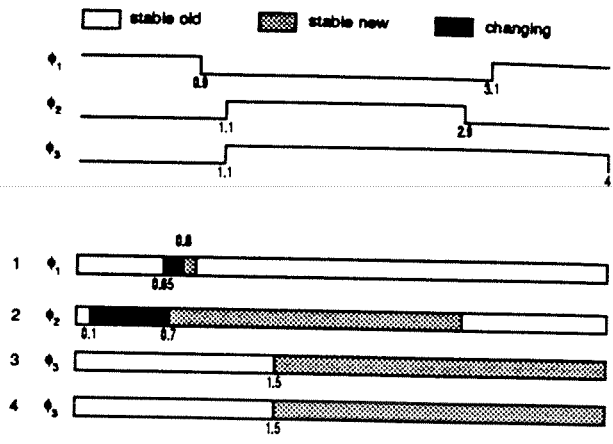


Figure 8: Three phase timing for the CPU.

4.2 FD-TLM Simulator

We have developed a simulator which solves for the behavior of circuits in terms of the three-dimensional electromagnetic field distribution. A variable-mesh finite-difference transmission-line-matrix (FD-TLM) method [6] is used which performs a full-wave solution of Maxwell's equations. In this approach, all couplings between interconnects, both capacitive and inductive are automatically included, and in addition, radiation effects are taken into account. The simulator also allows lumped models of nonlinear devices (e.g. transistors) to be included self-consistently. The FD-TLM program is very computationally intensive (currently it runs on a Cray Y-MP) so that it will be used in developing simpler models and to check the accuracy of critical results determined by lower level simulators. The TLM method uses the analogy between voltage and current on a three-dimensional mesh of transmission lines to the electric and magnetic fields in Maxwell's equations. The method proceeds by determining the scattering of impulses arriving at each node of the mesh at discrete time intervals. The resulting equations are recast in the form of finite difference equations in order to improve the efficiency by reducing the number of operations per step and the memory required compared to original TLM method.

Topics which will be explored by this method include simulation of chip-to-package pad bonds, interconnects including terminating loads, effects of imperfect matching of interconnects due to multiple fan-out and crosstalk for critical signals. In general, FD-TLM simulation will concentrate on parts of the circuit which cannot easily be modeled by lumped circuit simulators because of three-dimensional effects or long-range distributed interactions.

4.3 Timing Analysis

The performance levels for which our design aims, require that the processor module be viewed as a single integrated entity from the perspective of timing. Elaborate interchip signaling protocols reduce performance by increasing the delays at chip-crossings, but they simplify the task of the systems integrator who typically needs to combine off-the-shelf chips. In our design the interchip interfaces can be customized to improve performance (for example we use native GaAs levels).

The performance gains that are possible, from treating the MCM as a single logic circuit, can only be realized through use of accurate timing analysis techniques. We have developed two new timing analysis tools, $checkT_c$ and $minT_c$, for this purpose [10]. The first tool, $checkT_c$, is a timing verifier which examines a circuit to see if it satisfies a specified clock schedule, and reports setup and hold time violations. The second tool, $minT_c$, is a clocking optimizer which determines the optimal clock schedule (i.e. the schedule with the minimum cycle time) that satisfies all timing constraints for a given circuit. The optimization problem is formulated as a sequence of linear programs derived from the propagation and synchronization constraints in the design. Both tools are based on a new timing model of synchronous digital circuits which is: 1) general enough to handle arbitrary multi-phase clocking; 2) complete, in the sense that it captures signal propagation along short as well as long paths in the logic; and 3) extensible to make it relatively easy to incorporate "complex" latching structures (including the common level-sensitive D-latch).

Figure 8 shows a portion of the output from $minT_c$ when it was used to minimize the clock cycle for a portion of the processor data path (see Fig. 5). There are three clock

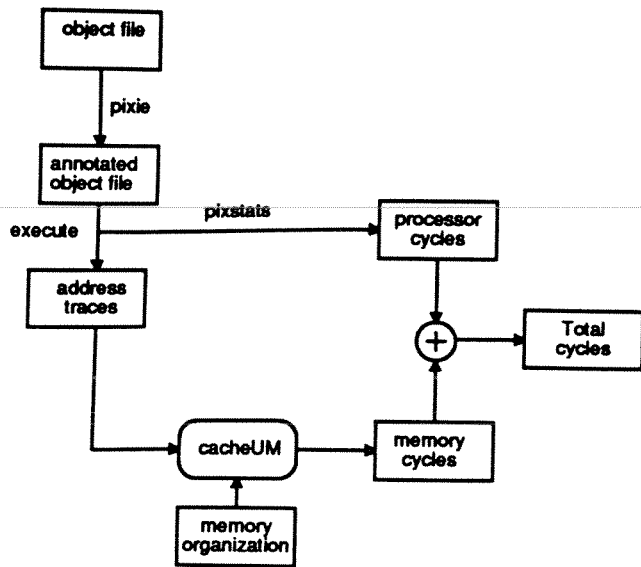


Figure 9: Cache simulation with `cacheUM`.

phases, shown at the top of the figure. More precisely, there are two non-overlapping phases (ϕ_1 and ϕ_2), and a derived third phase (ϕ_3) that is used solely to preset the sense amplifiers of the register file (see Sec. 3.1). The signals at four latches are also shown at the bottom of the figure. The corresponding latches are marked in Fig. 5.

4.4 `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 in Section 3.2.

Simulation using `cacheUM` operates as shown in Fig. 9. MIPS object code is annotated by `pixie`⁵ [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

⁵`pixie` is a trademark of the MIPS Corp.

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 stalls cycles caused by executing multi-cycle instructions such as integer multiply, integer divide, and floating-point instructions. The number of extra cycles is provided by `pixstats`⁶. 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 eighty 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 workload of a high performance workstation used in a technical environment. Each application is executed 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]. The composite CPI value is calculated 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 in millions it took to execute them, 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 in Section 3.2. 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 working sets, to share a 1 M-byte secondary cache without flushing each other's entries entirely between context switches.

In Table 3 we compare the performance of the micro-supercomputer with an R3000-based computer, the M/2000⁷. The M/2000 has 64 K-byte primary instruction and data caches, a four entry write buffer to main memory, and a 25 MHz clock. The cycle times differ by a factor of ten, whereas the TPI figures differ by a factor of 9.2.

⁶`pixstats` is a trademark of the MIPS Corp.

⁷M/2000 is a trademark of the MIPS Corp.

Name	Insts(M)	Cycles(M)	PI-miss(%)	PD-miss(%)	SI-miss(%)	SD-miss(%)	CPI
Sdiff	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
dhystone02	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
LFK(12)	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	1810.4	2672.6	1.80	6.88	2.13	5.47	1.48

Table 2: Benchmark performance results.

Machine	CPI	Cycle time(ns)	TPI	MIPS
2000	1.36	40	54.4	18.4
micro-super	1.48	4	5.92	168.9

Performance comparison between an M/2000 and micro-supercomputer.

Performance is due to the slightly poorer memory performance of the micro-supercomputer. However, the fact that the micro-supercomputer is able to increase the clock cycle by a factor of ten for a nominal reduction in architectural efficiency (measured by CPI) supports the design choice of using a cache subassembly for the cache.

5 Conclusions

The design considerations for a prototype micro-supercomputer, and have shown that in order to achieve the performance goals it is necessary to use an integrated design approach in which technology, architecture, and packaging are considered simultaneously. The use of GaAs DCFL technology which has high speed, high level of integration and high yield is an important factor in achieving the desired performance. Multi-chip packaging must be used to achieve the needed performance, and careful partitioning of the processes on the chip is required to minimize the number of chips on the critical path. The use of CAD tools is critical to the design 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. By coordinating these different aspects we believe that it is possible to achieve a global optimization of the design and build a system which will meet the specifications which we have described.

Acknowledgements

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 Corp. and Mentor Graphics Corp. 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 U.S. 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 Publishers, San Mateo, CA, 1990.
- [2] G. Kane, *MIPS RISC Architecture*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1988.
- [3] J. A. Dykstra, *High-Speed Microprocessor Design with Gallium Arsenide Very Large Scale Integrated*

- Digital Circuits*, Ph. D. Dissertation, University of Michigan, 1990.
- [4] H. B. Bakoglu, *Circuits, Interconnects and Packaging for VLSI*, Addison-Wesley, 1990.
- [5] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "check T_c and min T_c : Timing Verification and Optimal Clocking of Synchronous Digital Circuits," *Proc. of ICCAD-90*, (to appear).
- [6] R. H. Voelker and R. J. Lomax, "A Finite-Difference Transmission Line Matrix Method Incorporating a Nonlinear Device Model", *IEEE Trans. Microwave Theory Tech.*, Vol. 38, No. 3, March 1990, pp. 302-312.
- [7] Y. C. Lee et al., "Internal Thermal Resistance of a Multi-Chip Packaging Design for VLSI Based Systems", *IEEE Trans. Components, Hybrids, Manuf. Technol.*, Vol. 12, No. 2, June 1989, pp. 163-169.
- [8] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Co., Reading, MA, 1985.
- [9] J. R. Goodman, "Coherency for Multiprocessor Virtual Address Caches," *Proc. 2nd Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, October 1987, pp. 72-81.
- [10] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and Design of Latch-Controlled Synchronous Digital Circuits," *Proc. 27-th ACM/IEEE Design Automation Conf.*, (to appear).
- [11] *RISCompiler Languages Programmer's Guide*, MIPS Computer Systems, Inc., December 1988.
- [12] A. Agarwal, *Analysis of Cache Performance for Operating Systems and Multiprogramming*, Kluwer Academic Publishers, 1988.