GaAs RISC PROCESSORS

R. B. Brown, P. Barker, A. Chandna, T. R. Huff, A. I. Kayssi, R. J. Lomax, T. N. Mudge,
D. Nagle, K. A. Sakallah, P. J. Sherhart, R. Uhlig, and M. Upton

Solid State Electronics Laboratory and
Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, Michigan USA 48109-2122

ABSTRACT

A simplified version of a RISC microprocessor has been implemented with E/D MESFET DCFL in the Vitesse HGaAs II process. This chip was designed to drive the development of digital GaAs design automation tools. The processor architecture was modified to fit DCFL technology. The 60,500-transistor circuit executes a set of 29 basic instructions. It dissipates 11 W and operates at over 100 MHz. The RISC processor chip set being developed in this project is helping to identify the challenges and opportunities of VLSI GaAs.

INTRODUCTION

Competition among manufacturers of computing equipment is intense. No place is this more evident than in the engineering workstation market where in the past decade, computing power has grown by two orders of magnitude while price has decreased by an order of magnitude. According to Dataquest, over half a million workstations were sold in the world in 1991; this volume is expected to grow at a compound annual rate of 65.2% to 6.5 million units in 1996 [1]. The performance of these systems is largely dependent on the speed and interconnections of a few chips which comprise integer and floating-point processors, primary cache, and memory management. In the most successful of these computers, RISC (reduced instruction-set computer) architecture processors have displaced the CPUs of earlier generations.

The merits of various technologies for implementing these chip sets will continue to be debated as rapid advances are made in CMOS, BiCMOS, and ECL, as well as in GaAs bipolar and FET technologies. The workstation application demands not only ever-increasing system throughput, but also air cooling. The power-dissipation limits imposed by air cooling argue against using most bipolar logic families. In FET technologies, system performance suffers so much from chip-crossing delays that a process must support high levels of integration to be competitive. Defect problems related to epitaxial layers reduce the yields of MODFETs compared to MESFET and JFET technologies, which are more mature, and available from a number of commercial foundries. The one GaAs logic family which has demonstrated the integration level required by RISC chip-sets is DCFL (direct-coupled FET logic) [2, 3].

In the immediate future, then, the most promising technologies for high-end workstations will be CMOS, BiCMOS and GaAs DCFL. The silicon processes definitely have advantages in terms of circuit density, flexibility, and compatibility with other system components. Silicon also has momentum because of its market share and a good record of meeting performance goals; for digital GaAs to make a place for itself in the microprocessor market, it will have to demonstrate a system-level advantage of 2-to-1 over CMOS in speed or power-delay product. On the other hand, FET processes in GaAs are very simple, keeping tooling and processing costs low and resulting in good yields on large circuits. (Some BiCMOS processes have more than twice as many mask steps as DCFL.) The high electron mobility of GaAs is responsible for the good speed of these devices, but equally important is the fact that GaAs achieves its high mobility at low electric fields. This means that good speed can be realized with lower power supply voltages. The excellent low voltage operation of GaAs will be increasingly important as the market for battery-operated communications equipment and portable and wireless-networked computers grows. Unlike CMOS or BiCMOS, DCFL has small logic swings, so power dissipation is a weaker function of clock frequency. At high clock frequencies, the power dissipation advantage of CMOS vanishes, and DCFL is more power efficient. Finally, though the functional density is lower, the cost of prototype circuits in DCFL, on a per-square-millimeter basis, is now comparable to that of state-of-the-art CMOS; the cost of production circuits is still greater in DCFL.

In this paper we describe a prototype RISC-architecture microprocessor named Aurora, which our group designed and tested to explore the potential of VLSI circuits in GaAs. This is the first of three CPUs to be implemented in our DARPA-funded program. Since advanced design automation tools would leverage the whole project, we initially focused resources on extending commercial CAD tools to GaAs DCFL. This first CPU was designed to drive the development of these tools, and at the same time yield performance statistics on major circuit blocks. It was demonstrated only on a digital tester. The second CPU has been designed and fabricated; architectural, circuit, and performance details of this circuit will be reported at a later date. This chip will be prototyped with GaAs primary cache, CMOS secondary cache, CMOS FPGA-based memory management, and a bus interface on a DECStation turbobus. The third CPU prototype will include primary cache on-chip. It will be packaged on an MCM with secondary cache, a GaAs floating point accelerator (FPA), and a GaAs memory management unit (MMU), and will run a conventional unix environment.

AURORA ARCHITECTURE

From the block diagram (see Fig. 1) Aurora is seen to be a minimal RISC implementation, consisting of a 3-port register file, an arithmetic-logic unit, an instruction-decode / control section, a program counter section, and the necessary latches and multiplexers to implement a 5-stage pipeline, such as is used in several of the common RISC architectures. The pipeline stages are instruction fetch (I), register file access (R), ALU (A), data cache read / write (D), and register file write-back (W). A set of 29 instructions was selected for execution in Aurora: full-word load and store, 10 3-operand ALU operations, 8 immediate instructions, and 9 of the branch and jump instructions.

The issues in high-performance GaAs design are the same as in any other technology. The higher the switching speed, though, the more critical these issues become; delays from chip-crossings, on-chip interconnect, and loading effects can easily overwhelm the switching speed of gates [4]. Furthermore, DCFL circuits are less efficient than CMOS because of their limitations related to fan-in, fan-out, pass gates, stacked transistors, dynamic circuits and complex gates.

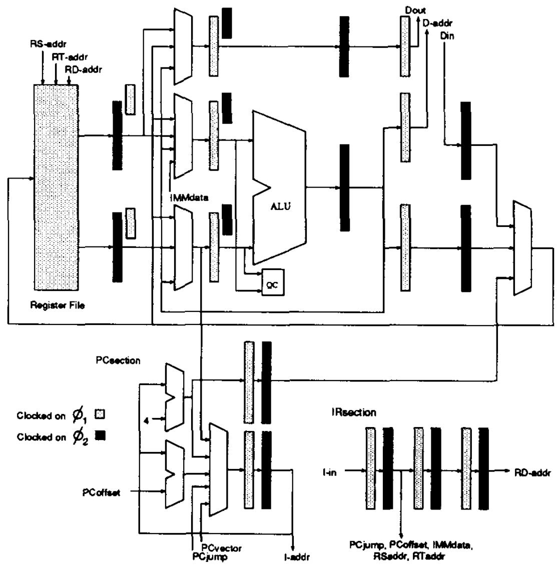

Fig. 1: Block diagram of Aurora architecture.

We have considered the strengths and limitations of DCFL technology not only in circuit design, but also at the architectural level [5]. The architecture was based on the MIPS R3000 [6], but many changes were made to better fit GaAs DCFL [7]. For example:

- Shared memory data and address buses were separated. A GaAs CPU needs all of the bus bandwidth for just the instruction cache.
- Pipeline timing was changed (see Fig. 2) from one clock cycle per stage to half cycles in the R and W stages to achieve a one-instruction branch delay.
- The single-level cache was changed to a two-level system with a direct-mapped primary cache [8, 9].
- The translation look-aside buffer was moved off-chip and out of the critical path to primary cache. The ability to squash instructions has to be added to accommodate this change.
- Integer multiply and divide functions were pushed into the floating-point accelerator.
- Byte operations were not implemented.
- And the data format option (big or little endianess) was dropped.

In addition, some features which will be included in later versions were eliminated to simplify the hardware in this first CPU: shifting, traps, system calls and cache control. The architecture of our system continues to change as we come to better understand DCFL design and the demands of high-performance processors.

FUNCTIONAL BLOCKS

Although most of the chip was not optimized for speed, significant effort was spent on two of the most critical elements of the chip, the adder and the register file.

The register file latch uses a 6-transistor RAM cell for data storage. A conservative register file readout design was chosen based on multiplexers, rather than a denser sense amplifier design. Using multiplexers minimized the design risk by keeping the entire register file readout in the digital domain. The 32 registers are selected using a 3-deep multiplexer tree. The first level of the tree

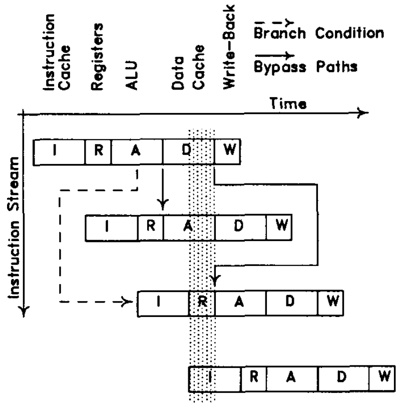

Fig. 2: Aurora pipeline representation, showing (shaded area) activity of pipeline during ϕ_1 clock.

selects between 4 latches using the 2 low-order bits of the register address. The next level of the tree selects between 4 first level muxes using the next 2 bits of the address. Finally, the register file output is selected using a 2-input multiplexer and the final bit of the register address. A register file write is performed by decoding the write address into 31 write lines, one for each address. Register 0 is hardwired to always contain 0.

The adder design is based on the approach developed by Ling [10] to take advantage of the wire-OR capability of ECL. The Ling adder carry signal is easier to generate and simpler to propagate than that of conventional adders [11]; this benefit also accrues in a GaAs DCFL implementation. The simpler carry is not without cost, however; the sum logic becomes more complicated. The added complexity in the sum generation can be hidden using a carry select method. In our implementation, the first level carry signals are generated in 3-bit groups rather than the typical 4-bit groups because of the limited fan-in capability of the DCFL gates. The second level carry signals are calculated in groups of 9 except for the highest-order group, which looks over 6 bits. Such an adder has 10 levels of logic, compared to 14 levels in a conventional 4-bit CLA approach.

The control consists of separate blocks for each of the five pipeline stages. A behavioral Verilog description (i.e., assignment and case statements) for each block was translated into input for Finesse, a logic synthesis tool from Cascade Design Automation. Finesse creates an optimized multilevel gate representation using a technology-specific library to generate the corresponding layout. Overall, the number of transistors in the control is 1840, and the density is 954 transistors/mm^2.

The problem of overdriving gates on heavily-loaded DCFL lines, such as clocks and datapath control signals, was dealt with in this chip by clamping the line with a diode at the source. This approach is expensive in terms of power, but the power dissipation is virtually independent of frequency. (Better buffering techniques are used in our recent designs, but they do add a significant frequency-dependent component to power dissipation.) A manual attempt was made to equalize transistor and interconnect loads on the multilevel clock distribution tree.

DESIGN METHODOLOGY

To achieve the potential performance of DCFL, one must also address design methodology issues. Because of the difficulty of designing full-custom GaAs (compared to CMOS), the most common design style for large digital circuits has become the gate array, which shields the designer from many of the unpleasant details of DCFL design. Unfortunately, in doing so, it gives up much of the speed advantage of GaAs. Average interconnect length in gate arrays is several times that in an equivalent custom design, significantly increasing the capacitive load. Because FETs have comparatively low transconductance, increased load slows propagation times significantly [4]. Furthermore, gate arrays offer only coarse sizing of gates to match their loads. A gate-array design style reduces the impact of the MESFET's switching speed on system performance.

Our first RISC CPU was designed with a preliminary version of a GaAs circuit compiler which yields physical designs that are comparable to full-custom layout. These tools, developed with Cascade Design Automation, Bellevue, WA, now include design entry in Verilog, VHDL or EDIF (Synopsis and schematic interface); synthesis and optimization of circuits from functional descriptions; design-rule-driven generation of GaAs cell layout; SPICE-based simulation models; back annotation of both gate delays and interconnect RC delays; automatic power rail sizing; and automatic, timing-driven buffer sizing, block placement and routing to minimize critical paths. Use of synthesized layout methods usually represents some compromise, but there is an opportunity with these performance-driven CAD tools to actually improve the speed of VLSI designs over that of hand crafted methodologies. Because these tools quickly implement physical layout and accurately identify the critical paths, it is practical to modify the design and recompile to meet performance goals. Another important advantage of this design approach is that it allows designs to be easily reimplemented in new design rules, so they can take advantage of the latest processes. Such tools should have a significant enabling effect on the digital GaAs area. The fact that Aurora was designed by five graduate students in just five months, including much work on the CAD tools, underscores the power of the design methodology.

PERFORMANCE RESULTS

The Aurora floorplan and photomicrograph of the chip are shown in Figs. 3 and 4. It was implemented in the Vitesse HGaAs II process (1.2-μm drawn / 0.8-μm effective gate length), and includes 60,500 transistors. The chip was packaged in a 344-pin package which required a frame size of 12.2 x 7.9 mm. It uses 172 signal and 108 power pins, and dissipates 11 watts.

Testing was done on an HP82000 IC evaluation system configured with 160 200-MHz channels and 16 400-MHz channels. Test vectors for Aurora were created as C programs with embedded assembly code, which was compiled and then disassemled on a DEC workstation. The code was compiled without optimization so that the machine code is identical in organization to the assembly code, and can be identified and extracted. The machine code was used as a ROM file in Mentor's Quicksim to set up the registers and latches in the Aurora model for the desired test. At this point, Quicksim scanned out the desired data, which were translated into HP82000 test vectors using TSSI software. For speed testing, the scan-in and scan-out sections were expanded so that they could run slower than the speed test itself.

Aurora has one human design error (an instance of misapplied source follower buffers) which disables some output pins. This problem was discovered shortly after the design was submitted for fabrication; fortunately, the scan chain allows testing of the chip despite the error. There was also a bonding problem which made the scan chain invaluable. The chip was otherwise fully functional, and had a packaged yield on 24 prototypes of 16.7%.

Extensive functional testing of the register file was done using vectors generated as described above, in testflow programs on the HP82000. Table 1 summarizes the results of these low speed (3.33 MHz) tests. Using asymmetrical clocks, the four fully functional register files had cycle times (a write followed by a read) of 6.4 to 6.7 nS. HSPICE simulations of the register file, including approximations of interconnect length, predicted write and read times of 1.8 and 2.1 nS, or an optimum cycle time of 3.9 nS. Aurora did not include predecoding of instructions, so most of the difference in measured and predicted values can be accounted for by the instruction decode delay, which is added to the access time in this chip.

While the circuit topology for the 32-bit adder was optimized, in this version, buffers were not sized optimally. HSPICE predicted a 4.6-nS adder cycle time from an extracted netlist, including parasitics. Again, because instruction decode is added to the cycle time, it was impossible to precisely determine the speed of indivudual function blocks. ALU functionality was tested on 12 chips, including all four of the ones that passed the register file test. Of these, six chips were fully functional and had propagation delays of 6.2 to 8.0 nS, with the average time being 7.25 nS. When the instruction decode time is taken into account, the measured adder speed falls into the range of 4 to 5 nS.

The best chip overall (register file and ALU on the same chip with the same clock schedule for both blocks) operates at 137 MHz. This does not necessarily mean that all of the other circuitry on the chip would run at this speed. The bonding problems prevented speed tests of branch instructions, which we believe would have limited the speed. On the other hand, with a larger sample of parts, one could expect to find chips on which both the register file and ALU are fast.

Table 1: Summary of register file functional analysis.

Chip Status	Chips	Probable Error Source
Fully Functional	4	
Fewer than 10 random bit failures	5	Bit Cell
Same-bit failures in multiple registers	7	Read-Out
Complete failure of one or more registers	2	Decode
Failure of all registers	6	Global

SUMMARY

The RISC processor chip set being developed in this project is helping to identify the challenges and opportunities of VLSI GaAs. DCFL has achieved the integration levels and yields needed to implement this type of design. When the microarchitecture is adjusted to fit the technology and an efficient design methodology is employed, DCFL is a serious contender for large, high-performance digital designs.

ACKNOWLEDGMENT

This work has been supported by the U.S. Defense Advanced Research Projects Agency under DARPA/ARO Contract No. DAAL03-90-C-0028, and by the U.S. Army Research Office under Contract No. DAAL03-87-K-0007.

REFERENCES

[1] Laura Segervall, Personal Communication, *Worldwide Workstation Market*, Dataquest Inc., San Jose, CA, May 1992.

[2] *Foundry Design Manual – Version 5.0*, Vitesse Semiconductor Corporation, Camarillo, CA, 1991.

[3] "A Record for GaAs Integration," *ASIC:EDA Technologies for System Design*, February 1992, p. 18.

[4] R. B. Brown, A. Chandna, T. R. Huff, R. J. Lomax, T. N. Mudge, R. Oettel*, and M. Upton, "Compound Semiconductor Device Requirements for VLSI," *Proceedings of the 19th International Symposium on Gallium Arsenide and Related Compounds*, Karuizawa, Japan, September 28 – October 2, 1992.

[5] O. A. Olukotun, R. B. Brown, R. J. Lomax, T. N. Mudge and K. A. Sakallah, "Multilevel Optimization in the Design of a High-Performance GaAs Microcomputer," *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 5, May 1990, pp. 763–767.

[6] Gerry Kane, *MIPS RISC Architecture*, Prentice-Hall Inc., Englewood Cliffs, NJ., 1988.

[7] T. N. Mudge, et al., "The Design of a Micro-Supercomputer," *IEEE Computer Magazine*, Vol. 24, No. 1, January 1991, pp. 57–64.

[8] O. A. Olukotun, T. N. Mudge, and R. B. Brown, "Cache Architectures for a High-Performance GaAs Microprocessor," *Proceedings of the 18th International Symposium on Computer Architecture*, Toronto, May 27–30, 1991, pp. 138–147.

[9] Kunle Olukotun, Trevor Mudge, and Richard Brown, "Performance Optimization of Pipelined Primary Caches," *Proceedings of the 19th International Symposium on Computer Architecture*, Gold Coast, Australia, May 19–21, 1992, pp. 181-190.

[10] H. Ling, "High Speed Binary Adder," *IBM Journal of Research and Development*, 25(3), pp.156–166, May 1981.

[11] N. Quach and M. Flynn, "High-Speed Addition in CMOS," Technical Report CSL-TR-90-415, Stanford University, February 1990.

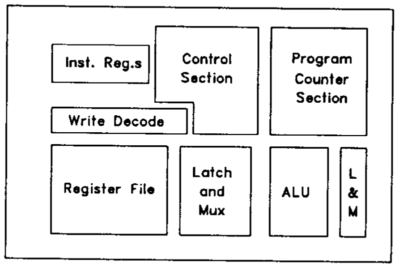

Fig. 3: Floorplan of the Aurora chip.

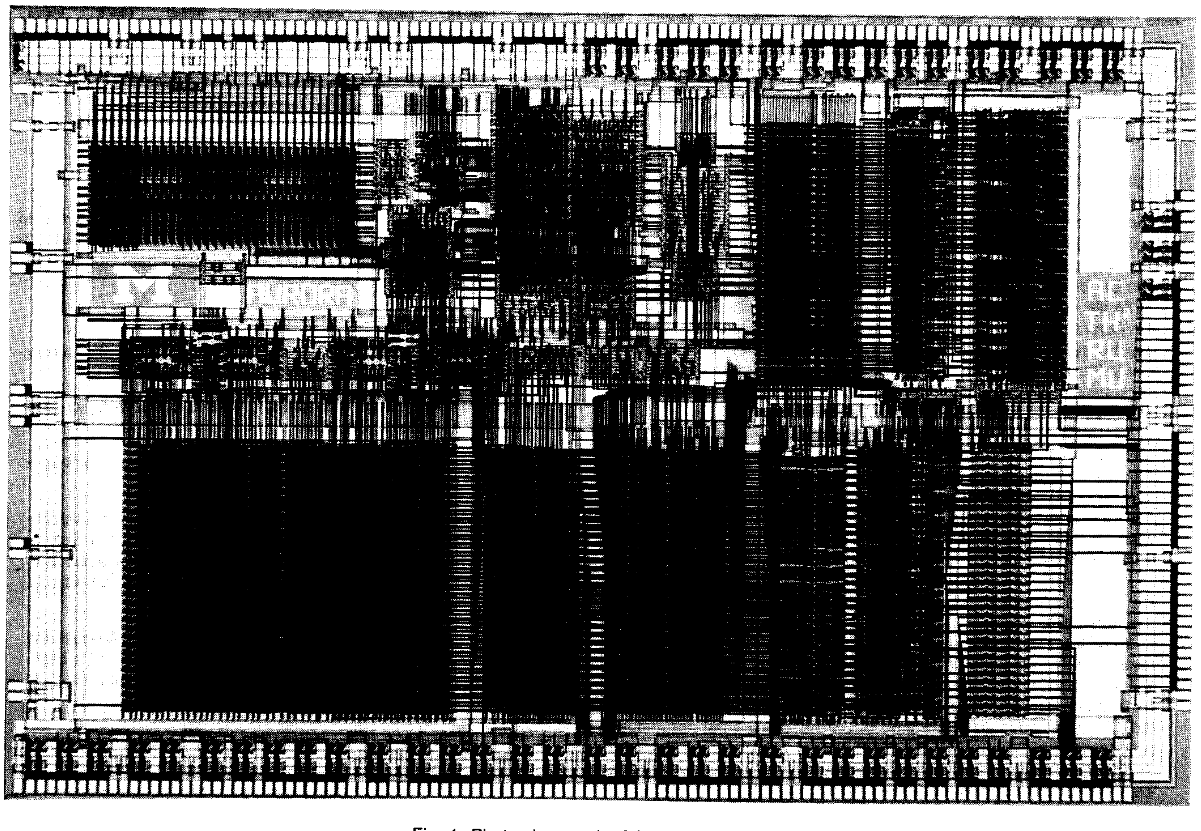

Fig. 4: Photomicrograph of Aurora RISC chip.