Gallium-Arsenide Process Evaluation Based on a RISC Microprocessor Example

Richard B. Brown, *Senior Member, IEEE,* Michael Upton, *Student Member, IEEE,* Ajay Chandna, *Student Member, IEEE,* Thomas R. Huff, *Student Member, IEEE,* Trevor N. Mudge, *Senior Member, IEEE,* and Richard E. Oettel, *Member, IEEE*

***Abstract*— This work evaluates the features of a gallium-arsenide E/D MESFET process in which a 32-b RISC microprocessor was implemented. The design methodology and architecture of this prototype CPU are described. The performance sensitivity of the microprocessor and other large circuit blocks to different process parameters is analyzed, and recommendations for future process features, circuit approaches, and layout styles are made. These recommendations are reflected in the design of a second microprocessor using a more advanced process that achieves much higher density and performance.**

I. Introduction

SINCE the introduction of GaAs circuits, high-performance digital systems have been considered a major potential application area for III/V technology. As is often the case in advanced technologies, GaAs device and process developers have sometimes emphasized ring-oscillator gate delay to the neglect of other characteristics important to VLSI circuits. The focus on gate delay led to overoptimistic predictions of the system-level performance advantage of GaAs over silicon. Early fabrication challenges, limited integration levels, and poor load-driving capability compared to bipolar technologies also hindered the acceptance of GaAs. Only recently has the promise for high-speed VLSI in gallium arsenide begun to be fulfilled, with direct-coupled FET logic (DCFL) circuits now being delivered in supercomputers [1]–[4], signal processors [5], [6], and telecommunication systems [7].

In terms of circuit density, flexibility, and compatibility with other system components, silicon logic families definitely have advantages over DCFL. On the other hand, FET processes in GaAs are very simple; few mask levels should lead to low tooling and processing costs, and to good yields. The high electron mobility of GaAs is responsible for the 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, giving DCFL a good power–delay product compared to other high-speed technologies. Unlike CMOS or BiCMOS, DCFL has small logic swings, so power dissipation is a weaker function of clock frequency. At high clock frequencies, DCFL is more power effficient than CMOS.

Manuscript received March 30, 1993; revised June 4, 1993. This work was supported by the U.S. Defense Advanced Research Projects Agency under DARPA/ARO Contract DAAL03-90-C-0028, and by the U.S. Army Research Office under Contract DAAL03-87-K-0007.

R. B. Brown, M. Upton, A. Chandna, T. R. Huff, and T. N. Mudge are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122.

R. E. Oettel is with Cascade Design Automation, Bellevue, WA 98006.

IEEE Log Number 9211376.

Our experience designing GaAs microprocessors has helped both to demonstrate the capabilities of DCFL in large circuits, and to clarify the device and process requirements of compound semiconductors for VLSI. In this paper we present an overview of the design methodology (Section II), and the architecture (Section III) of a prototype RISC processor which our group designed to explore these issues. We then discuss the strengths and liabilities of DCFL, through examples of large circuit blocks, including the microprocessor as a whole (Section IV). The performance sensitivity of these circuits to different design rules and process parameters is analyzed, and recommendations for future process features, circuit approaches, and layout styles are made. These recommendations are reflected in the design of a second microprocessor using a more advanced E/D MESFET process that achieves much higher density and performance.

II. Design Methodology

Our microprocessors have been designed with a GaAs circuit compiler [8] which produces layouts that have physical data paths organized as one would in a handcrafted design, minimizing chip area and total interconnect length compared to standard-cell- or array-based methodologies. We enter the design in a mixed behavioral/structural hardware description language (HDL) [9], which is one of several input formats available for the CAD tools. An interface package converts the HDL description into an internal netlist and partitions the design into structural and behavioral blocks. The structural blocks are implemented as data paths and the behavioral blocks are synthesized and implemented as standard cells. The tools provide design-rule portability, so that a given design can be evaluated in different rule sets or easily translated into a newer process.

Use of synthesized layout methods usually represents some compromise, but there is an opportunity with these CAD tools to actually improve the speed of VLSI designs over that of handcrafted methods. 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. The routers support multilevel interconnect, variable width signal routing, multiphase clock distribution, ground planes, and automatic power-rail sizing for IR drop and electromigration. The analysis capability includes a static timing analyzer that handles both single-phase and two-

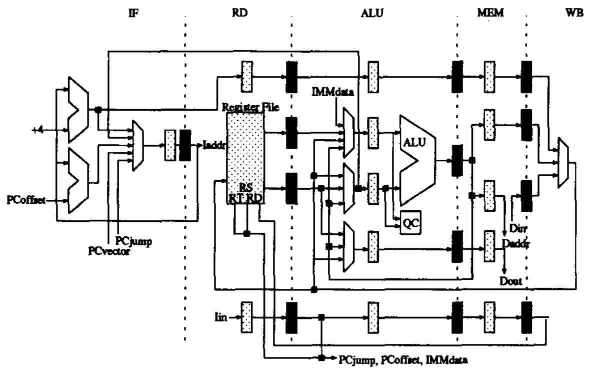


Fig. 1. Block diagram of baseline GaAs microprocessor architecture.

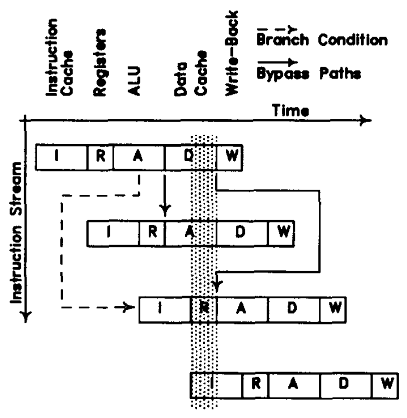


Fig. 2. Microprocessor pipeline representation, showing (shaded area) activity of pipeline during ϕ_1 clock.

phase clocks, and delay calculations that include interconnect *RC* delay. The CAD tools now include automatic performance-driven placement and buffer sizing, which we expect to further improve speed and power dissipation in our next generation of chips. Such tools should have a significant enabling effect on the digital GaAs area. The fact that our processors have been designed by four or five graduate students in less than six months, including much work on the CAD tools, underscores the power of the design methodology.

III. Microprocessor Architecture

Knowing that advanced design automation tools would leverage the whole project, we initially focused resources on developing the CAD environment for GaAs described above. Our 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 block diagram of Fig. 1 shows that this processor is a minimal RISC implementation, consisting of a three-port register file, an arithmetic-logic unit, an instruction-decode/control section, a program counter section, and the necessary latches and multiplexors to implement a five-stage pipeline, such as is used in several of the common RISC architectures [11]. The pipeline stages (see Fig. 2) 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 this CPU: full-word load and store, ten 3-operand ALU operations, eight immediate instructions, and nine of the branch and jump instructions.

In the design of this CPU, we considered the strengths and limitations of DCFL technology not only in circuit design, but also at the architectural level [10]. The architecture was based on a commercial instruction set architecture [11], but many changes were made to better fit GaAs DCFL [12]. For example:

- Shared memory data and address buses were separated. A GaAs CPU needs all of the bus bandwidth for just the instruction cache.
- The single-level cache was changed to a two-level system with a direct-mapped primary cache [13], [14].
- Integer multiply and divide functions were pushed into the floating-point accelerator (which has a parallel multiplier) to better utilize transistor resources; this improves performance.
- Byte operations were not implemented; this allows the use of simple word-based SRAM's without requiring a read–modify–write operation.
- And the data format option (big or little endian) was dropped; this is not a commercial processor, so compatibility is not an issue.

In addition, some features included in our more recent versions were eliminated to simplify the hardware in this first CPU: shifting, traps, system calls, and cache control.

Although the chip was not optimized for speed, significant effort was spent on two elements of the chip, the adder and the register file, which were expected to be on the critical path. The register file latch used a six-transistor RAM cell for data storage. A conservative register-file readout design was chosen based on multiplexors, rather than a denser sense-amplifier design. Using multiplexors minimized the design risk by keeping the entire register file readout in the digital domain. The 32 registers are selected using a three-deep multiplexor tree. The first level of the tree selects between four latches using the two low-order bits of the register address. The next level of the tree selects between 4 first-level muxes using the next 2 b of the address. Finally, the register file output is selected using a two-input multiplexor 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 [15] 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 [16]; 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-b groups rather than the typical 4-b groups because of the limited fan-in capability of the DCFL gates. The second level carry signals are calculated in groups of nine except for the

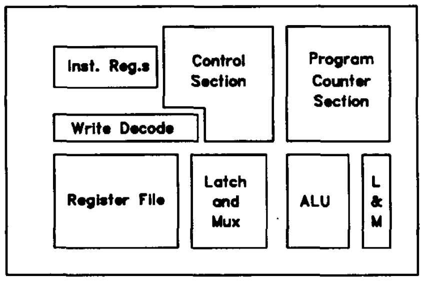


Fig. 3. Floorplan of the microprocessor chip.

highest-order group, which looks over 6 b. Such an adder has 10 levels of logic, compared to 14 levels in a conventional 4-b CLA approach.

The control consists of separate blocks for each of the five pipeline stages. A behavioral description for each of these blocks was translated into the CAD tools for logic synthesis [8]. The tools create an optimized multilevel gate representation using a technology-specific library [17] to generate the corresponding layout. In this case, the circuit is implemented in NOR-only DCFL logic, with fan-in typically limited to 4 (in one case it was as high as 7), and a maximum fan-out of 10. The number of transistors in the control is 1840, and the density is 954 transistors/mm^2.

The MESFET's Schottky-diode gates and high source resistance can cause an overdriving condition when a large DCFL buffer is used to achieve a short delay with a highly capacitive load. A large current is needed to charge the line quickly, but it may be too large for the static current sinking capability of the gates on the line. In extreme cases, gate current flows not only through the source, but also out the drain, increasing the output voltage to a logic ONE, when a logic ZERO was desired. Large buffers on this chip, such as those driving clocks and data-path control signals, had diode clamps at their outputs to avoid overdriving the gates on these lines. This approach is expensive in terms of power, but the power dissipation is virtually independent of frequency. (Superbuffers are used instead in our recent designs. They are more power efficient, but they do make power dissipation a stronger function of frequency.) The interconnect loading and number of loads on the multilevel clock distribution tree were optimized manually.

A floorplan and photomicrograph of the CPU are shown in Figs. 3 and 4. This 60 500-transistor circuit was implemented in a process having a 1.2μm drawn (0.8-μm effective) gate length. The chip was packaged in a 344-pin package that required a frame size of 12.2 × 7.9 mm. It uses 172 signal and 108 power pins, and dissipates 11 W.

The CPU was found to have one human design error (an instance of misapplied source-follower buffers) that disables some output pins. This problem was discovered shortly after the design was submitted for fabrication; fortunately, the scan chain allowed testing of the chip despite the error. There was also a bonding problem that 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 was done on the register file [18]. Table I summarizes the results of these tests. Using

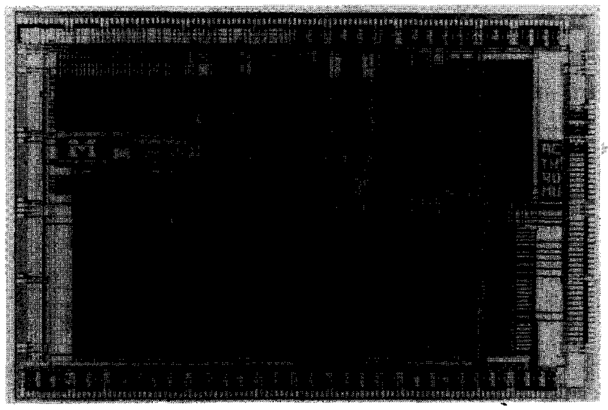


Fig. 4. Photomicrograph of GaAs RISC microprocessor.

TABLE I
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

asymmetrical clocks, the four fully functional register files had cycle times (a write followed by a read) of 6.4 to 6.7 ns. This chip did not include predecoding of instructions, so these times include both instruction decode and read/write access times.

While the circuit topology for the 32-b adder was optimized, in this first version of the CPU, buffers were not sized optimally. ALU functionality was tested on 12 chips, including all four chips that passed the register file test. Of these, six ALU's were fully functional and had propagation delays of 6.2 to 8.0 ns, with the average time being 7.25 ns. Again, the instruction decode time accounts for some of this delay.

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.

IV. PROCESS ANALYSIS

Because of its simplicity, the ring oscillator is often used as a performance monitor during process development. Although ring oscillators do provide valuable information, ring-oscillator data must be used with caution to avoid overestimating the system-level performance that can be achieved. To demonstrate this effect, we performed SPICE simulations of ring oscillators in two different GaAs DCFL processes. Each process was simulated with three different loads: one driven gate, four driven gates, and four driven gates plus 3 mm of on-

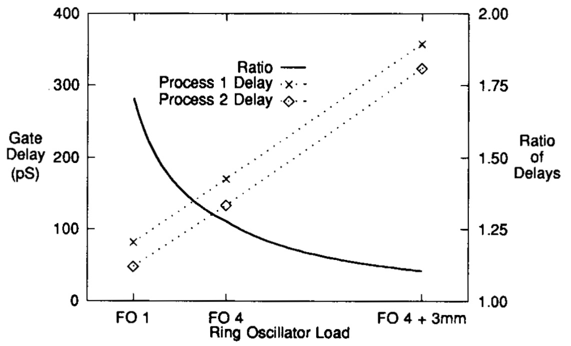


Fig. 5. Ring-oscillator performance for two different GaAs processes and three different load conditions. Gate delay is shown on the left axis and the ratio of delays on the right axis. The load position for a fan-out of 4 with 3 mm of wire is placed arbitrarily.

chip interconnect. The results (Fig. 5) demonstrate that in this case, lightly loaded ring-oscillator speeds can be deceptive. While the ratio of gate delays is 1.7 : 1 with a light load, in more realistic circuits where a gate drives multiple loads and interconnect, the ratio of delays is 1.1 : 1. Though the speed advantage in terms of gate delay is constant (the slopes of delay versus load are similar), the ratio of loaded delays is much smaller.

Intrinsic transistor switching time is certainly important for high-performance digital circuits, but overemphasis on this parameter can obscure other features of a technology which will determine its viability. All of the parameters of a semiconductor process are interdependent, and device and logic family characteristics are intimately related. Among the important features of a digital circuit process are integration level, yield, power dissipation, noise margins, interconnectability, load-driving characteristics, and availability of design automation tools. One would like to optimize every desirable parameter, but the parameters often present conflicts (such as speed versus noise margin) that require trade-offs to be made. Many of the parameters in this optimization have minimum requirements, though, below which digital circuits will not be competitive, no matter how attractive other features, such as switching speed, may be.

A. Integration Level

The first issue to be considered is integration level. The "package delay" associated with getting signals through an output buffer, off-chip interconnect, and an input buffer can account for a large percentage of the clock cycle time in high-performance systems, even when the most advanced packaging is used. For example, Kayssi *et al.*'s [19] simulations of our microprocessor, flip-chip mounted on a multichip-module (MCM) with a 4K-word instruction cache, show that the MCM delay is 45% of the total clock cycle. When the cache size is increased to 8K words, the clock period must be lengthened, and the MCM delay increases to 55% of the clock cycle. These percentages would be even higher with other packaging schemes. The package delay means that a slower technology which has high enough integration levels to keep the critical path on one chip can outperform a faster technology which

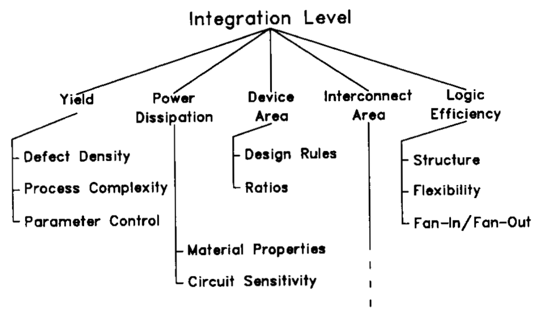


Fig. 6. Process parameter interdependencies.

has to have chip crossings in the critical path. Pipelining, advanced packaging schemes, and judicious partitioning can partially ameliorate the problem.

Integration level, in turn (see Fig. 6), is dependent on yield, power dissipation, logic style efficiency, active device area, and interconnect density. Yield is a function of material defect density, process complexity, and other factors which influence the level of parameter control that can be maintained. For applications that require air cooling, power dissipation becomes the integration-limiting parameter for many high-speed technologies. Because of the variation in transistor efficiency from one logic family to another, a true comparison of integration level between technologies would have to be done at the functional level. For example, though DCFL has only $n + 1$ transistors per gate compared to $2n$ transistors in a complementary CMOS gate, we have found in analyzing many random logic blocks, such as control circuits for the microprocessor pipeline stages, that DCFL typically requires two-thirds more gates (and logic levels) per function because it has low fan-in and fan-out, requires more buffering, provides only limited use of pass gates, and does not support complex gates or dynamic circuits.

The area occupied by active devices is a function of all of the design rules. Transistor area (the result of gate, source/drain, contact, and isolation design rules) is of prime importance in determining the density of RAM's, but it is less important than interconnect dimensions in determining the size of logic circuits composed of data paths and random logic. This is illustrated in Table II, a comparison of 8×8 Booth-encoded array multipliers implemented as data paths by our CAD tools in three DCFL processes, which have drawn gate lengths in the ratio shown. To make the results reflect differences in design rules, rather than number of interconnect layers, all of these circuits were routed in gate metal plus three interconnect levels, but with ground distributed on the top routing level instead of on the fourth level of metal available in two of the processes. As seen in the table, layout area is a much stronger function of interconnect dimensions than of gate length. Even more striking is the difference in total routing area, which directly affects interconnect capacitance.

B. Interconnect

The importance of interconnect in a VLSI process cannot be overstated. The switching delay τ for any logic family

TABLE II
COMPARISON OF 8×8 MULTIPLIERS IN THREE DCFL PROCESSES. ALL PARAMETERS ARE NORMALIZED

	Gate Metal	Metal 1	Metal 2	Metal 3	Total Layout Area	Total Routing Area
Process A	1.00	1.00	1.00	1.00	1.00	1.00
Process B	0.90	0.60	0.50	0.28	0.49	0.21
Process C	0.50	0.97	1.11	1.43	0.97	0.82

is related to the amount of charge at the output of a logic gate that must be supplied or removed to change states, and to the current available to effect this change of state: $\tau \propto C\Delta V/I$. Sensitivity to parasitic loading varies with process and logic family. In any FET technology, this is the dominant delay mechanism; it calls for small logic swings, high transconductance, and low-capacitance loads.

Most of the capacitive load comes from interconnect. Of primary importance is keeping the circuit area as small as possible to minimize wire length; this reduces both parasitic capacitance and time-of-flight for signals. Routing capacitance is minimized by using enough levels of interconnect, narrower lines, larger separation between interconnect layers, and lower dielectric-constant insulators. The effect of narrowing the separation between lines is not immediately obvious; while it reduces the circuit area, it does increase horizontal line-to-line capacitance. However, the total-routing-area data shown in Table II make a strong case for reducing interconnect spacing to the fabrication limits.

Design methodology also has a major effect on interconnect capacitance. Because of the difficulty of designing full-custom GaAs (compared to CMOS), the most common design method 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 FET's have comparatively low transconductance, increased load slows propagation times significantly. Furthermore, gate arrays offer only coarse sizing of gates to match their loads. To help quantify the efficiency of different design methodologies, we mapped the 8×8 array multiplier of Table II onto a sea-of-gates array provided by one of the three foundries; 100% gate utilization was assumed. The full implementation with our GaAs circuit compiler occupies only 63% of the area taken by the raw gates required in the array. Realistic cell utilization in the array would amplify this difference significantly.

Using the microprocessor design as a benchmark, we were able to evaluate the features of DCFL processes [20]. The importance of minimizing interconnect capacitance is illustrated by Figs. 7–9, which show the effects of reducing unloaded gate delay or capacitive loading on three critical paths in our microprocessor. The logic paths in these plots are from the branch logic, adder, and register file. The sensitivities of these effects vary among the paths simulated, but the plots show clearly that performance is dominated by interconnect loading, and therefore, reducing interconnect capacitance would be as effective at increasing circuit speed as would reducing intrinsic gate delay. The closest results are for the branch logic, where

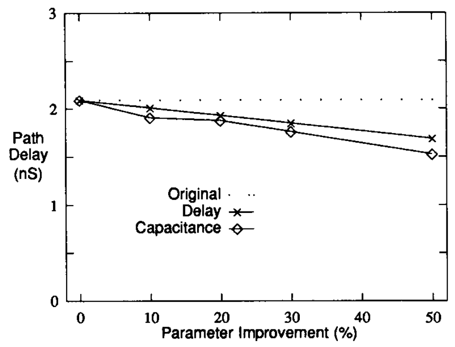


Fig. 7. Branch logic critical path sensitivity to gate delay and capacitive loading. The horizontal axis shows a percentage decrease in capacitive load and in gate delay.

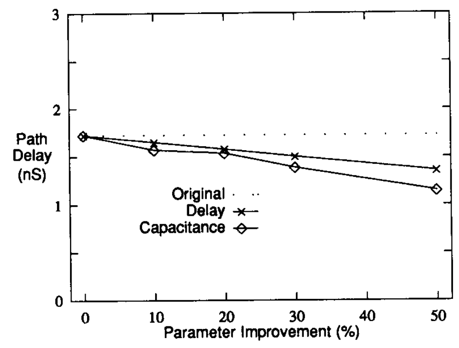


Fig. 8. ALU path sensitivity to delay and capacitive loading. The horizontal axis shows a percentage decrease in capacitive load and in gate delay.

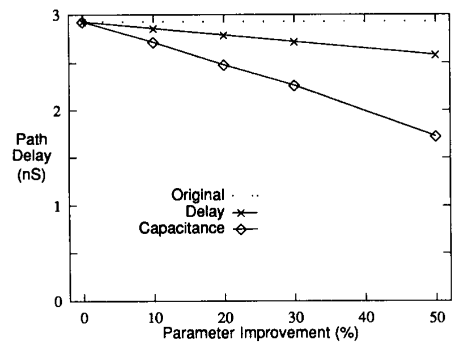


Fig. 9. Register-file critical path sensitivity to delay and capacitive loading. The horizontal axis shows a percentage decrease in capacitive load and in gate delay.

a 50% reduction in capacitance has a 40% greater effect than a similar reduction in unloaded gate delay. The biggest difference is in the register file, where capacitance reduction has a 248% greater effect.

The importance of having enough layers of interconnect merits further illustration. The CPU described above was implemented in a three-metal process. A second version of the CPU has been implemented in a process having four levels of metal and 0.6-μm effective channel length; this CPU includes more functionality than the first version, and more effort was expended optimizing it for speed [21]. Table III shows the improvement in density that we have achieved by moving

TABLE III
DENSITY COMPARISON BETWEEN THREE-METAL AND FOUR-METAL PROCESSES

Circuit	3-Metal Transistor Count	3-Metal Density (Trans./mm^2)	4-Metal Transistor Count	4-Metal Density (Trans./mm^2)
Largest Control Block	582	1067	516	1364
Register File	21,910	2014	23,278	4253
CPU	60,500	540	160,000	1475

from the three-metal to the four-metal process. In the four-metal process, we use gate metal and metal 1 for wiring inside of leaf cells, and metals 1, 2, and 3 for data paths, standard cell blocks, and global routing. Metal 4 is a ground plane, and V_{dd} is distributed on metal 3. Of course, geometric design rule changes between the processes and other factors, noted below, also affect the density. The control blocks are different circuits (bypass logic in one case and stall logic in the other), but they are about the same size, and both are implemented in standard cells using the same logic synthesis tool [8]. The register files in Table III are both 32-word × 32-b, three-port, tree-decoded, pass-gate latch implementations, which differ only in buffering.

The density numbers for the CPU's include all of the unoccupied space in the pad frame—there is actually more of it in the four-metal version. Some of the increase in density is due to the inclusion of additional memory structures for a small on-chip instruction cache on the four-metal CPU. But aside from this, the four-metal version of the CPU is still about 2.4 times denser. Analysis of the density in these processes is facilitated by the CAD tools, which allow efficient implementation of circuits using various combinations of the process features. We found that half of the improvement is due to the additional interconnect layer; improved circuit structures and layout techniques incorporated into our newer CAD tools account for another 35%; and the remaining 15% of improvement results from smaller line widths in the newer process.

Adding too many wiring layers would result in diminishing improvement in density. Large DCFL circuits need more interconnect layers though than CMOS because one layer is used as a ground plane.

C. Memory-Related Issues

To avoid the chip-crossing delay mentioned above, many digital systems will require embedded memory. Our own GaAs SRAM work is leading toward on-chip primary cache. Memory must be dense and power efficient if it is to be embedded. The need to integrate memory with large logic circuits adds to the list of process requirements in a technology for digital circuits.

For example, in SRAM's, chip size and power are strong functions of leakage current. Though much less attention has been focused on minimizing leakage currents than on increasing transconductance, leakage currents are as important to performance. If too many memory cells are connected to a bit line, the leakage current through the pass transistors connected to unselected memory cells (about 100 nA/b) could corrupt the data of a selected memory cell (about 20 μA). The total leakage on a bit line should be an order of magnitude smaller than the active current, so the number of bits that can be safely connected to a column is limited to 32. This constraint requires that a significant portion of the total RAM area be devoted to sense amplifiers and write circuitry [22]. Table IV shows how SRAM area would decrease if leakage currents could be reduced to allow more memory cells per column, thereby amortizing the column support circuitry over more bits. As can be seen, for this design at 32 b/column only 70.6% of the total chip area is consumed by the memory cells. A reduction in leakage current by one order of magnitude would increase the percentage of area occupied by the memory cells to 92% of the total area.

TABLE IV
EFFECT OF REDUCING LEAKAGE CURRENTS ON AREA OF 1K×8 SRAM

	Number of Bits / Column 32	64	128	256	512
Normalized SRAM Area	1.00	0.87	0.80	0.77	0.75
Cell Area Percentage of Total Area	70.6	81.6	88.4	92.1	93.8

In any technology, the pull-up of a static RAM cell should provide just enough current to offset the leakage current of the pull-down devices. (Leakage currents, therefore, also set the lower limit for cell power.) In conventional GaAs DCFL processes, long minimum-width depletion transistors are used to keep this current small. The characteristics of these devices present an area/power trade-off. For example, in our SRAM, the highest impedance standard-threshold depletion transistor that fits in a 400 μm^2 cell provides much more current than is needed to offset the leakage currents. As the area of the cell is decreased, the pull-up length must be decreased, increasing the power. Fig. 10 shows the effect of varying the pull-up length (cell size) on power dissipation. This plot includes curves for a digital process pull-up transistor, a more positive-threshold depletion transistor, and a resistor load. The load curve for resistors was constructed assuming they could be located above the remaining four transistors, adding no additional area. As seen in the figure, resistor loads are invaluable to SRAM designs.

V. SUMMARY

It is the performance of compound semiconductors in systems, not ring-oscillator speeds, that will dictate their future in digital applications. We have developed automatic, design-rule portable, physical design tools for DCFL circuits, which allow easy comparison of different processes. Using a simple RISC microprocessor as a benchmark, we have been able to evaluate the effect of various design rules and process features on system performance. Without high integration levels, the speed of compound semiconductors is lost to chip-crossing delays. Circuit performance can be improved faster by improving interconnect than by improving device switching speed. Embedded memory will be necessary in the highest performance systems, so digital processes need to provide memory-specific features.

Our experience designing and testing large digital circuits and implementing layout generators which support various

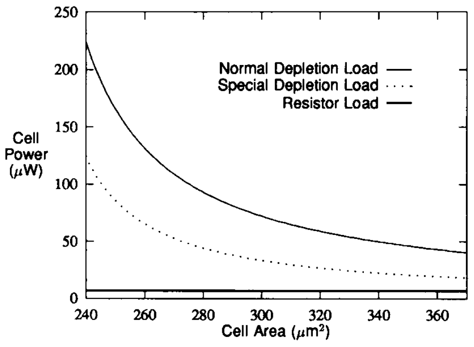
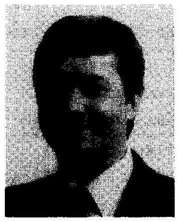


Fig. 10. SRAM cell power vs. cell size for three load devices: a normal depletion load, a special RAM depletion load having a more positive threshold, and a resistor load.

GaAs processes has helped clarify the size, power, and system performance dependencies on device and process characteristics.

REFERENCES

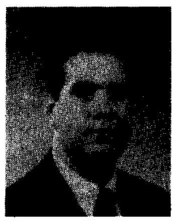
[1] "The GaAs supercomputer challenge," *Electron. Eng. Times*, pp. 39–44, Jan 20, 1992.
[2] *VPP500 Vector Parallel Processor* (BP0009-1M), Fujitsu Limited, Tokyo, Japan, Sept. 1992.
[3] D. Kiefer and J. Heightley, "CRAY3: A GaAs implemented supercomputer system," in *Proc. 1987 IEEE GaAs IC Symp.*, pp. 3–6.
[4] R. Alverson *et al.*, "The Tera computer system," presented at the 1990 ACM Int. Conf. Supercomputing, June 11–15, 1990.
[5] M. Rocchi, *High-Speed Digital IC Technologies.* Norwood, MA: Artech House, 1990.
[6] V. Peterson, "Applications of GaAs IC's in instruments," in *GaAs IC Symp. Tech. Dig.*, Nov. 6–9, 1988, pp. 191–194.
[7] *1992 Product Data Book*, Vitesse Semiconductor Corp, Camarillo, CA, 1992.
[8] *EPOCH Designer's Handbook*, Cascade Design Automation, Bellevue, WA, 1992.
[9] D. E. Thomas and P. Moorby, *The Verilog Hardware Description Language.* Dordrecht, The Netherlands: Klewer, 1991.
[10] 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 J. Solid-State Circuits*, vol. 26, no. 5, pp. 763–767, May 1990.
[11] G. Kane, *MIPS RISC Architecture.* Englewood Cliffs, NJ: Prentice-Hall, 1988.
[12] T. N. Mudge *et al.*, "The design of a micro-supercomputer," *IEEE Computer*, vol. 24, no. 1, pp. 57–64, Jan. 1991.
[13] O. A. Olukotun, T. N. Mudge, and R. B. Brown, "Cache architectures for a high-performance GaAs microprocessor," in *Proc. 18th Int. Symp. Computer Architecture* (Toronto), May 27–30, 1991, pp. 138–147.
[14] K. Olukotun, T. Mudge, and R. Brown, "Performance optimization of pipelined primary caches," in *Proc. 19th Int. Symp. Computer Architecture* (Gold Coast, Australia), May 19–21, 1992, pp. 181–190.
[15] H. Ling, "High speed binary adder," *IBM J. Res. Develop.*, vol. 25, no. 3, pp. 156–166, May 1981.
[16] N. Quach and M. Flynn, "High-speed addition in CMOS," Tech. Rep. CSL-TR-90-415, Stanford Univ., Stanford, CA, Feb. 1990.
[17] *Foundry Design Manual—Version 5.0*, Vitesse Semiconductor Corp., Camarillo, CA, 1991.
[18] R. B. Brown, *et al.*, "GaAs RISC processors," in *GaAs IC Symp. Tech. Dig.*, Oct. 4–7, 1992, pp. 81–84.
[19] A. I. Kayssi, *et al.*, "Impact of MCM's on system performance optimization," in *Proc. 1992 IEEE Int. Symp. Circuits Syst.*, vol. 2 (San Diego, CA), May 1992, pp. 919–922.
[20] R. B. Brown, *et al.*, "Compound semiconductor device requirements for VLSI," in *Proc. 19th Int. Symp. Gallium Arsenide and Related Compounds* (Karuizawa, Japan), Sept. 28–Oct. 2, 1992.
[21] M. Upton, *et al.*, "A 160,000 transistor GaAs microprocessor," in *ISSCC Tech. Dig.* (San Francisco, CA), Feb. 24–26, 1993, pp. 92–93.
[22] A. Chandna and R. B. Brown, "A 32kb GaAs SRAM with electronically programmable redundancy," in *Proc. 1993 Symp. Research Integrated Syst.* (Seattle, WA), Mar. 15–16, 1993, pp. 155–167.



Richard B. Brown (S'74–M'76–SM'91) received the B.S. (with highest honors) and M.S. degrees in electrical engineering (computer emphasis) from Brigham Young University, Provo, UT, in 1976. He received the electrical engineering Ph.D. (solid-state/VLSI area) from the University of Utah, Salt Lake City, in 1985. His dissertation work included development of a custom MOS fabrication process and integration of digital and analog circuitry to form a novel solid-state chemical sensor.

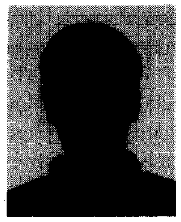
From 1976 to 1981 he worked in computer design as Vice-President of Engineering at Holman Industries, Oakdale, CA, and then as Manager of Computer Development at Cardinal Industries, Webb City, MO. In September 1985 he joined the faculty of the University of Michigan Department of Electrical Engineering and Computer Science, Ann Arbor. He has been involved in shaping the VLSI program and introducing a uniform set of electronic CAD tools into the curriculum. He has taught solid-state devices and digital electronics, introduction to semiconductor device theory, and both introductory and advanced VLSI design courses. For the past seven years, he has worked in VLSI digital GaAs circuits and high-performance computing systems. He holds five patents, and consults in the areas of solid-state devices and electronic design automation tools.

Dr. Brown has served as advisor to the local student chapter of IEEE.



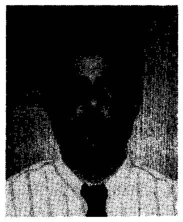
Michael Upton received the B.S.E.E. and M.S.E.E. degrees from the University of Washington in 1984 and 1990, respectively. He began work on his Ph.D. degree at the University of Michigan, Ann Arbor, in 1991 and is currently a Ph.D. candidate studying VLSI circuits and high-performance computing.

From 1984 to 1990 he worked for Seattle Silicon Corporation developing computer-aided design tools for integrated circuit placement and routing.



Ajay Chandna (S'93) was born in Windsor, Ont., Canada, in 1967. He received the B.A.Sc. degree in electrical engineering from the University of Windsor, Ont., Canada, in 1989 and the M.Sc. degree from the California Institute of Technology, Pasadena, in 1990. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering at the University of Michigan in Ann Arbor. Since 1989 he has been involved in various VLSI design projects including an analog/digital neural network, optoelectronic integrated circuits, smoothing networks, and a GaAs microprocessor. His current research interests include high-speed GaAs SRAM design and CAD tool development.

Mr. Chandna is a recipient of the Canadian Natural Science and Engineering Research Council 1967 Scholarship.



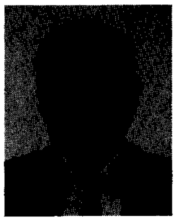
Thomas R. Huff (S'87) received the B.S.E.E. degree from the Department of Electrical Engineering at the University of Michigan, Ann Arbor, in 1989 and is currently working toward the Ph.D. degree at University of Michigan. Since 1989 he has been involved in various VLSI projects including the design of a CMOS math coprocessor and two GaAs microprocessors. His current research interests include architectural and circuit issues in implementing high-performance floating-point hardware in GaAs.

Mr. Huff is a recipient of an ARO University Research Initiative Fellowship.

Trevor N. Mudge (S'74–M'77–SM'84) received the B.S. degree in cybernetics from the University of Reading, England, in 1969, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively. While at the University of Illinois he participated in the design of several high-performance computers, and did research in computer architecture.

Since 1977 he has been on the faculty of the University of Michigan, Ann Arbor, where he has taught classes on logic design, CAD, computer architecture, and programming languages. He is presently a Professor of Electrical Engineering and Computer Science and the Director of the Advanced Computer Architecture Laboratory, a group of eight faculty and 60 graduate research assistants. He is author of more than 100 papers on computer architecture, programming languages, VLSI design, and computer vision, and he holds a patent in computer aided design of VLSI circuits. His major research, at present, is the design and construction of a high-performance GaAs "microsupercomputer." In addition to his position as a faculty member, he is a consultant for several computer companies in the areas of architecture and CAD.

Dr. Mudge is a member of the ACM, a member of the IEE, and a member of the British Computer Society. He is currently Associate Editor for *ACM Computing Surveys,* Subject Area Editor for the *Journal of Parallel and Distributed Computing,* and a member of the Editorial board of *IEEE Parallel & Distributed Technology.*



Richard E. Oettel (S'58–M'61) received the B.S. degree, cum laude, and the M.S. degree, both in electrical engineering, from the University of Washington in 1959 and 1964, respectively.

He is a founder of Cascade Design Automation Corporation (formerly Seattle Silicon Corporation) in Bellevue, WA, and is its Chief Scientist. In this office he is responsible for advanced projects, new technology development, and Cascade's University program. He is a co-inventor of Cascade's design-rule-driven IC design methodology. Prior to founding Seattle Silicon, he managed a semiconductor laboratory at the Boeing Aerospace Company where he was responsible for the development of over 200 semiconductor devices and integrated circuits in support of various research projects.