A GaAs Micro-Supercomputer: Rationale and Design

By

R.B. Brown, J.A. Dykstra, T.N. Mudge,
and R. Milano

CSE-TR-42-90

THE UNIVERSITY OF MICHIGAN
COMPUTER SCIENCE AND ENGINEERING DIVISION
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
Room 3402, EECS Building
Ann Arbor, Michigan 48109-2122
USA

A GaAs Micro-Supercomputer: Rationale and Design

R.B. Brown, J.A. Dykstra, T.N. Mudge
EECS Department
University of Michigan
Ann Arbor, MI 48109-2122

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

January 10, 1990

Abstract[1]

This report describes a project to design and build a micro-supercomputer which will be capable of a sustained performance of over 100 MIPS. Simulated performance on the Livermore Loops is over 20 MFLOPS, which equals the performance of a Cray 1S. The micro-supercomputer, though, will be significantly faster on non-vectorizable and integer code. The CPU implements the MIPS architecture in GaAs using E/D MESFETs. This combination marries GaAs speed with a system and chip architecture tailored to the advantages and constraints of the technology. We believe that previous work in digital GaAs has fallen short of the performance which is possible because of limitations of early GaAs technologies, inadequate packaging techniques, and suboptimal design styles. An accurate appraisal of digital GaAs capabilities is needed to allow meaningful comparisons with alternative technologies.

An existing instructions-set architecture (ISA) was chosen to allow us to focus on the implementation and technology aspects of constructing a micro-supercomputer. Even with the ISA defined, there are still important architectural issues to be solved, particularly the design of a memory subsystem capable of supplying instructions and data to the CPU at peak rates of 1 G-byte per second. This paper, though, focuses on the CPU chip and the issues involved in the design and layout of GaAs VLSI. We present data from Vitesse that supports the view that DCFL logic implemented in GaAS E/D MESFETs is a strong contender for very fast VLSI logic. In particular, it is simple, relatively low power, and has a high enough yield to support the design and production of VLSI parts with over 60,000 transistors, enough to support a RISC-style CPU. We further show that the disadvantages of DCFL, low fan-in and fan-out, low noise margins, susceptibility to common-mode noise, and lack of CAD tools, can be overcome. The use of superbuffers and new bus designs can increase fan-in and fan-out and increase noise margins. The use of 3-level interconnect with large ground buses, which is supported by the Vitesse technology, moderates the intrusion of common-mode noise. To optimize density and minimize parasitic loading so as to realize the full potential of MESFET speed, custom design is essential but extremely time consuming without adequate CAD tools. To fill this gap we are developing a GaAs compiler in cooperation with Seattle Silicon Corp.

The integer processor (IP) is presented as an example to show that GaAs VLSI is possible with present technology. Central to the CPU is a register file of thirty-two 32-bit registers with two read and one write port that can be accessed simultaneously. The ALU supports 32-bit additive and multiplicative operations as well as the usual complement of logical and load / store operations. Instruction execution is pipelined into 5 stages. The design of the datapath has been simulated with V-SPICE, a SPICE variant developed by Vitesse for their GaAs process. The simulation has been performed for ambient temperatures over the range 0° to 70°C to verify performance over temperature-dependent threshold shifts. These simulations show that a 4 nS clock will not violate any timing constraints. The integer processor will be combined on a multichip substrate, with a floating-point unit, cache, and memory management chips to form a micro-supercomputer.

Parts of the microprocessor have been fabricated already and are being tested. Results from these tests will be included in future reports.

[1] This work is supported by the Army Research Office through Contract DAAL03-86-K-0007, and by the National Science Foundation through Grant MIP-8802771.

A GaAs Micro-Supercomputer: Rationale and Design[1]

R. B. Brown, J. A. Dykstra, T. N. Mudge
EECS Department
University of Michigan
Ann Arbor, MI 48109-2122

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

1 Introduction

Technology selection for high-performance digital systems is a decision of fundamental importance, and the relative merits of alternative technologies are the subject of impassioned debate in the semiconductor industry. Due in part to the challenge from competing approaches, CMOS, BiCMOS, ECL, and GaAs MESFET technologies have all made rapid performance improvements over the past few years. Accurate appraisals of the capabilities of these alternatives are important because they affect industrial and governmental research funding levels. Funding levels, in turn, determine the rate of future advances in each logic family.

GaAs technologies are interesting candidates for high-performance computing because of the high electron mobility of GaAs, and because their semi-insulating substrates reduce stray capacitance. Pioneering work has been done in the areas of technology [1,2,3], signal-processor design [4], and microprocessor design [5]. Efforts to implement GaAs microprocessors have had a variety of principal goals, including low-power and radiation hardness (McDonnell Douglas [6]), control of threshold voltage and high integration level (TI [7]), and high-speed at modest integration levels (RCA [8], Vitesse [9]). With the experience of these initial projects, the merits of GaAs are now better understood, and early over-optimistic speculation has given way to a common view that the performance margin of gallium arsenide over silicon will be between 2 to 3 times. We believe that previous work in digital GaAs has fallen short of the performance which is possible because of limitations of early GaAs technologies, inadequate packaging techniques, and suboptimal design styles. A convincing demonstration of the potential of GaAs VLSI for implementing very high-speed systems remains to be made.

As a vehicle for exploring GaAs VLSI and demonstrating its capabilities, we are designing a micro-supercomputer, built around a 250 MHz GaAs microprocessor. Simulations on a wide range of realistic benchmarks show that the system will be capable of a sustained performance of more than 100 MIPS. Development of this system will combine the following ingredients: 1) Technology (GaAs

[1]This work is supported by the Army Research Office through Contract DAAL03-86-K-0007, and by the National Science Foundation through Grant MIP-8802771.

E/D DCFL from Vitesse Semiconductor Corp., Camarillo, CA, and advanced multi-chip packaging to reduce chip-crossing delays); 2) Architecture (MIPS instruction-set architecture from MIPS Computer Systems of Sunnyvale, California, and multilevel cache); 3) Software (operating systems and optimizing compilers from MIPS); and 4) Design Aids (performance-driven circuit synthesis, and advanced modeling tools). A prototype system will be assembled using our processor module (integer processor (IP), floating-point unit (FPU), cache, and memory management) in a high-performance chassis (primary memory, backing store, and peripheral subsystem) from MIPS Computer Systems.

This paper will focus on the integer processor chip and the issues involved in the design and layout of GaAs VLSI. To provide context, we include brief descriptions of the technology, design methodologies and architecture.

2 Technology

Technology includes both semiconductor and packaging issues. 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 line-driving capabilities are not as good as those of 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 processor module will be assembled using flip-chip mounting on a multi-chip substrate having high-density interconnect. Modeling of these packages is being done with the Finite-Difference Transmission Line Matrix Method [10] in parallel with high-frequency characterization of prototype packages. The need of using such packaging techniques for very fast systems is becoming widely accepted. The choice of semiconductor technology, on the other hand, remains a subject of debate.

To be suitable for high-speed VLSI, a logic family must provide low propagation delay, low power per gate, low dynamic switching energy, very high gate density, and high yields. Other circuit characteristics are also important. For example, noise tolerance must be adequate even in the face of common-mode-coupled switching noise and electromagnetic-coupled crosstalk, which are more troublesome at high switching speeds. GaAs material and manufacturing problems have been controlled to the point that gate-array ASICs and standard SSI, MSI and SRAM components, are readily available, and are being designed into mainframe computers. Yields in gallium arsenide which would allow VLSI integration levels are, however, recent developments.

Many GaAs logic families have been developed or proposed, including: depletion-only logic (BFL, FL, FFFL, CCFL, SDFL, CDFL, and SCFL), complementary JFET logic (CJFET), bipolar logic

(CML and HBT I^2L), and unipolar logic (DCFL with resistive, saturated-enhancement, or depletion loads, buffered DCFL, FFL, SBFL, and SCFL). The depletion-mode families have excellent gate speed, but have unacceptable density due to higher power, the added complexity of output level shifters and multiple power supplies. JFET logic operates with little power, but speed is not one of its virtues. The bipolar CML (current-mode logic) is a promising long-term option for very high-speed systems, but integration level will likely lag that of MESFET processes as ECL lags CMOS. Power dissipation, which limits integration level, is a challenge for any bipolar logic family.

We selected E/D DCFL (Enhancement/Depletion Direct Coupled FET Logic), which is similar to silicon E/D NMOS, as the basic logic family for this project because of its good speed (130 pS gate delays), simplicity (few devices per gate), and low power. 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 Power-Delay Product of E/D DCFL is clearly superior to that of other MESFET-transistor logic families. Speed of FET circuits increases more rapidly with scaling than does that of bipolar circuits, so as geometries are scaled toward 0.1 μm [11], the speed differential between FET and bipolar devices will be reduced. We expect gallium arsenide and silicon FET logic to continue to increase in speed with improving process technology, but GaAs will retain the speed advantage due to its material properties.

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 because of the Schottky gates, and therefore small noise margins. 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. Substrate material is 4-inch diameter, undoped, semi-insulating GaAs. A direct step-on-wafer exposure system is used for pattern definition; this system uses 5× image reduction with a maximum field of 14 mm square. The process includes three active and two passive elements: enhancement- and depletion-mode MESFETs, Schottky-barrier diodes, resistors and capacitors. It includes 33 major steps, requiring eleven masks, and supports variable gate lengths, with a minimum length of 1.0 μm and no limit on maximum length. 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 three-layer Al 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

Circuit [2]		Transistors	Size (mils)	Die / Wafer	$Y_{P\ ave}$	Best Wafer	First Fabbed
VS12G	1K SRAM	10000	85 x 70	860	38	65	12/87
VSC1500	Gate Array	8000	137 x 159	320	71	94	5/87
VSC4500	Gate Array	16000	280 x 160	160	53	80	11/87
VSC15K	Gate Array	60000	280 x 335	80	31	61	11/88

[2]Gate array implementations are for customer circuits having 80–90% utilization.

Table 1: Vitesse Semiconductor E/D DCFL yield data.

mask layers of current ECL processes.

The end-to-end yield for an IC process is a product of fabrication yield, Y_F, probe yield, Y_P, assembly yield, Y_A, and final test yield, Y_{FT}. Each yield term represents the ratio of apparently good circuits passing on to the next processing step, to the number of circuits started (into fabrication, probe, assembly or final test, respectively). For current semiconductor processing equipment and methods, Y_F = 0.9 – 1.0, and $Y_A \times Y_{FT}$ = 0.85 – 0.99. Most failures are detected at probe test, Y_P, which verifies full circuit functionality over specified dc, power supply, and input signal variations. The probe yield for chips fabricated with the Vitesse process is shown in Table 1. $Y_{P\ ave}$ in this figure is the average probe yield for all production lots produced in the last 12 months; all die sites are included in the calculation. "Best wafer" yield is the best wafer produced over the last 12 months. Figure 1 shows the dramatic improvement in end-to-end yield as as function of time, with data shown for three gate-array parts.

Process design for high yield, and improvements in starting material and process control have made VLSI integration levels practical in GaAs DCFL now, and yields will continue to improve. There are, however, 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.

The noise margin problem can be moderated by circuit and layout design. For example, super-buffers, with their improved high and low output levels, can be used on all long lines, and resistive voltage drops on ground lines can be ameliorated with wide ground distribution buses. These and other design approaches reduce the impact of process variations across a chip. Super-buffer advantages, unfortunately, are paid for in increased area and power, especially during switching. (Conventional DCFL gates are similar to current steering logic in that they steer a relatively constant load current either into the gate diodes or through the inverting transistor.) Both increased area and greater switching power exacerbate resistively coupled noise on ground. Wide ground traces also come with a price: achieving area-efficient layouts of large, complex chips with wide ground rails

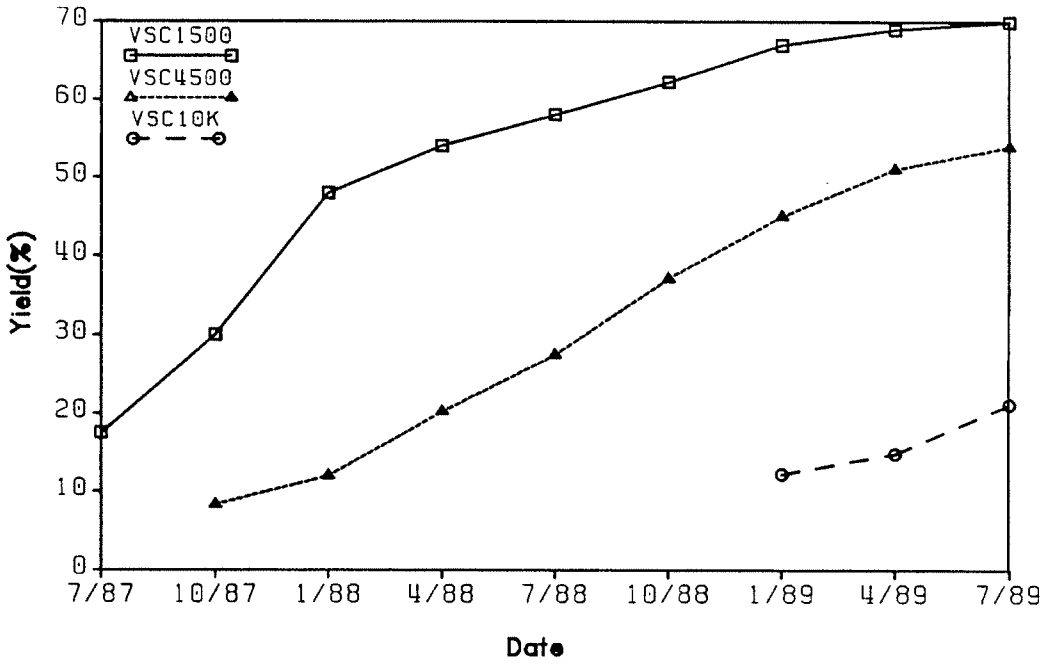

Figure 1: Vitesse functional probe yield.

requires three levels of metal interconnect. This third layer with large power traces, in turn, increases parasitic interconnect capacitance on signal lines below.

In summary, the level of integration becoming available through yield improvements in DCFL GaAs makes implementation of a 32-bit RISC processor on a single chip possible [12]. This is a critical integration level for reducing the number of chip crossings. While, DCFL is more efficient than ECL for VLSI, when compared to NMOS or CMOS, DCFL still presents the designer with annoying limitations.

3 Design Aids

To date, gallium arsenide ICs have been designed using layout tools developed for conventional silicon VLSI. While the use of many of these is appropriate, design of very fast VLSI circuits presents a need for some specialized design aids. With higher switching speeds, layout becomes much more important, as parasitics become the dominant source of delay. For circuits of low complexity, designs could be tailored by hand, but the speed potential of DCFL can be realized only at higher levels of integration. Optimization of a complex layout by hand for line length or crosstalk, for example, can be intractable.

The difficulty of designing full-custom GaAs (compared to NMOS and CMOS) has led to use of the gate array, which shields the designer from many of the aforementioned details, as the most common form of digital GaAs. Unfortunately, gate arrays give up much of the speed advantage of MESFETs. Propagation time in any FET technology is determined by transconductance of the active devices and parasitic loading. Average interconnect length in a gate array is several times as long as in an equivalent-size custom chip, increasing gate loading, crosstalk, and signal ringing. (This is true, to a lesser extent in standard cell design, which is also used for GaAs.) Furthermore, the area efficiency of gate-array and standard cell layout is much lower than that of custom design. Interconnect length grows with chip area. Since the integration level is lower than in CMOS, this loss of circuit density is more serious for GaAs. A dramatic illustration (in CMOS) of the difference in area efficiency of several methodologies is shown in [13]. A simple 16-bit RISC processor was implemented in the same design rules, using gate array, standard cell, and silicon compiler design tools, all from VLSI Technology, Inc. The ratio of resulting chip areas was approximately 3:2:1. Designing in these styles, with their large interconnect parasitics and low densities, means that the high speed of GaAs gates has reduced impact on system performance.

The CAD work in this project is improving circuit density by extending circuit compiler capability to GaAs. The tools, being developed in cooperation with Seattle Silicon Corp., include new functionality for generating high-performance integrated circuits and systems: synthesis of GaAs cells, including layout and simulation models; automatic buffer sizing to meet timing specifications; placement and routing which can optimize critical paths for propagation delay and noise coupling; automatic power rail design to reduce supply-coupled noise; and signal integrity control considering reflections, dispersion, crosstalk, and reduced noise margins characteristic of very high-speed systems. Use of synthesized layout methods represents some compromise, but there is an opportunity with these computer-aided capabilities to improve the performance of VLSI designs over that of hand crafted methodologies. Such tools will have a significant enabling effect on the area of digital GaAs.

4 Architectural Overview

Our selection of an existing instruction-set architecture allowed work in GaAs chip design to begin immediately, and avoided the need for us to develop operating system and compiler software. The resulting system will be general purpose, running a conventional UNIX environment and supporting standard programming languages and networking protocols. A large base of existing application software could benefit from availability of such a machine, which operates an order of magnitude faster than comparable present-day computers.

The reduced instruction set computer (RISC) design philosophy is well suited to GaAs implementation. The simple instruction set can be easily pipelined, leading to fast chips with low device counts. Low device count is particularly important to GaAs because of its comparatively lower integration level. Pipelining provides for smaller, and therefore faster, logic blocks. We chose the MIPS embodiment of RISC architecture because its optimizing compilers yield a low average number of cycles per instruction [14], its pipeline definition better meets our constraints, and its hardware multiply / divide capability improves execution time of many engineering application programs. (See Kane [15] for the MIPS instruction-set architecture (ISA) definition.)

Even with the instruction set defined, there are a number of architectural issues to be addressed including partitioning and microarchitecture of the processor, and design of a memory subsystem capable of delivering both instructions and data at a peak rate of 1 G-byte per second with latency of only one clock cycle. Increased latency would results in a corresponding increase in branch and load delays. The 4 nS cycle-time specification for our system rules out the possibility of time-multiplexing the translation look-aside buffer (TLB) or cache bus as is done on CMOS implementations. Furthermore, current levels of GaAs integration preclude implementing both the integer processor (IP) and TLB on a single chip, much less a chip with two TLBs. To meet these constraints, we have partitioned the design with two TLBs (on separate cache controller chips) and have modified the architecture to allow a delayed cache-tag match. We have also developed a two-level cache structure having split instruction / data primary cache, to be made of custom 1K $\times$ 8 GaAs SRAMs. As stated earlier, packaging of the processor module is critical because signals must cross chip boundaries on the paths between the IP and the cache, cache controller, and FPU.

Central to the design of the micro-supercomputer is the 32-bit integer processor, which performs all arithmetic, logical, addressing, and control functions required to execute the MIPS instruction set. The IP datapath is arranged to do operations in parallel, so each instruction is performed in stages. The pipestage sequence for our design is shown in Figure 2. During the I stage, an instruction is fetched from cache; one instruction is required every clock period unless there is a cache miss or other interrupt situation. In the R stage, the instruction is decoded and operands are read from the register file. ALU operations take place in the A stage, and the data cache is read or written during the D stage. Finally, during the W stage, results are written back to the register file. Every operand from memory (except immediates) must be temporarily stored in the register file before it can be used in a calculation. A two-phase non-overlapping clock scheme is used, with one cycle being 4 nS long. The A, I and D stages are one cycle long, but to accommodate the one instruction branch delay expected by the MIPS compiler, the R and W stages were made only 1 phase, or 2 nS long. Also shown in Figure 2 are two possible bypass paths, required to avoid hardware interlocks. Bypassing allows the

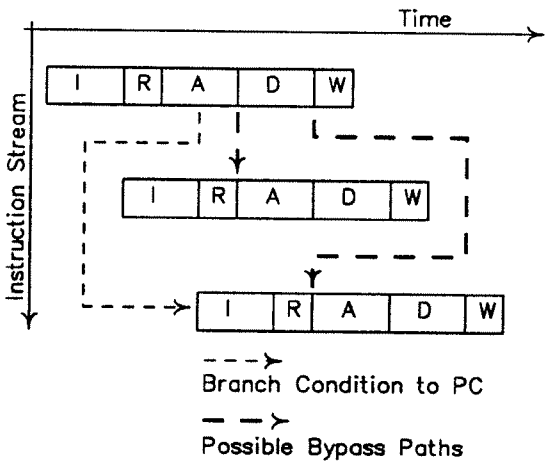

Figure 2: Pipeline Stages and Relative Timing

ALU to take operands from any pipe stage, even before the register file is updated with the new value.

The following section describes the design of the integer processor in more detail.

5 GaAs Integer Processor

Design decisions in the integer processor are based on the primary goals of executing the MIPS instruction set with a peak speed of 250 MHz, and of optimizing average throughput. The chip is designed in the Vitesse E/D MESFET technology to run over a temperature range of 0–70°C. The design is done primarily in static NOR logic, with NAND logic used sparingly. Layout for the IP control section is being done with the GaAs compiler described above; the datapath was implemented before the compiler was available, using the Mentor Graphics ChipGraph full-custom layout tool.

The basic structure of the chip is shown in Figure 3. The data path includes a register file (RF) with thirty-two 32-bit general purpose registers, an ALU, a shifter, and a multiply / divide unit (not shown). Note that a separate address adder and incrementer are required for the program counter section to comply with the instruction per clock-cycle philosophy. The data path and its control occupy the major portion of the chip area. The rest of the area is occupied by the exception handler (CP0), the program counter, and I/O interfaces.

5.1 Controller

The controller decodes instructions and distributes control signals to the register file, ALU, shifter, integer multiply / divide unit, load aligner, and to multiplexors which direct the flow of data

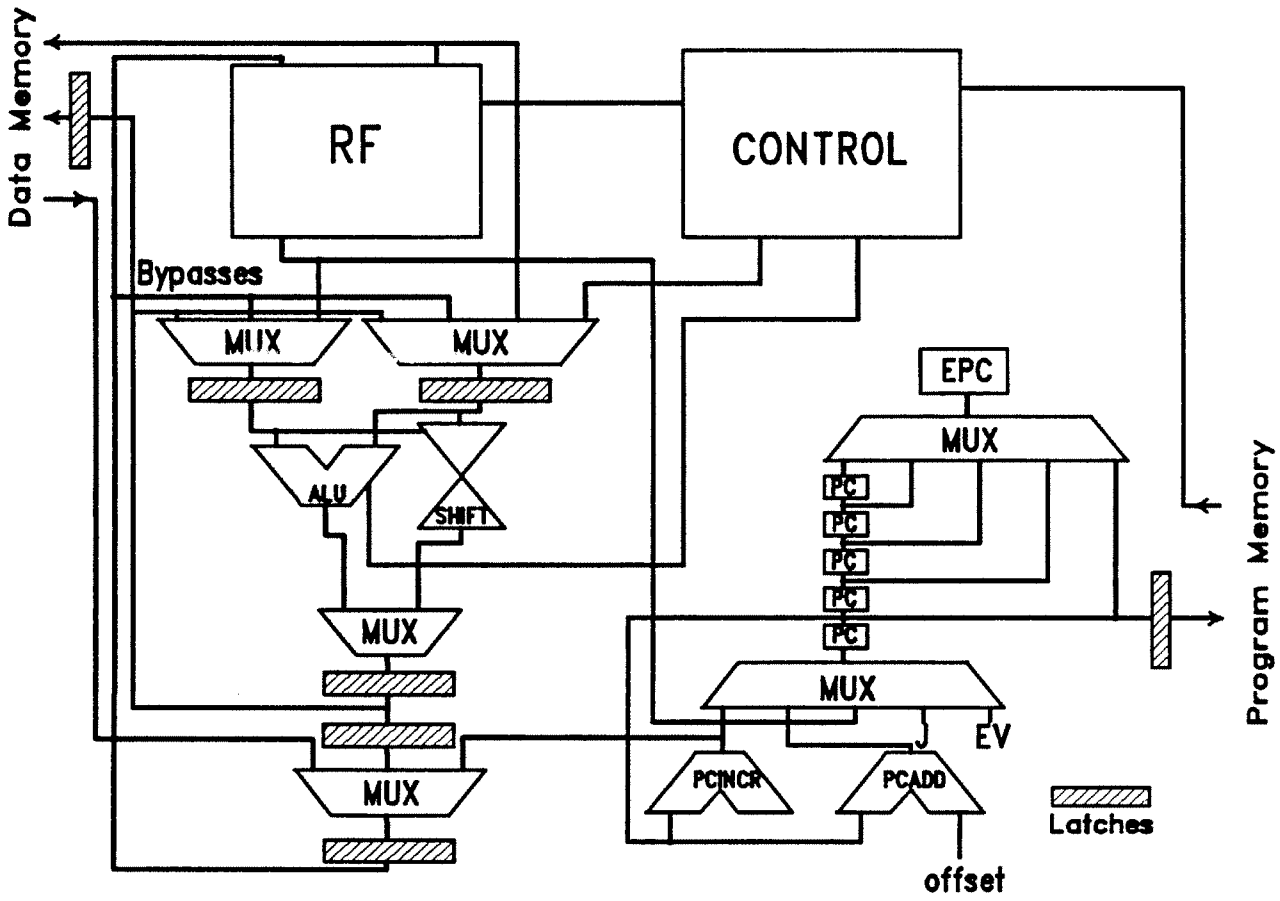

Figure 3: Register-transfer level block diagram of the GaAs Integer Processor.

Logic Block	Devices (Area Estimate)	Function	Delay	Power
Program Counter	2,400	Increment	2.3 nS	
		Add	2.5 nS	
Register File	16,213	Read	0.4 nS	1.53W
SRAM-like	(4.0 x 1.4 mm)	Read Set-Up	1.2 nS	
3-port, 32-Registers, 32-bits		Pre-Set	1.8 nS	
		Write	1.2 nS	
		Write Set-Up	0.8 nS	
Arithmetic Logic Unit	3413	Add	2.8 nS	0.67W
Binary look-ahead carry	(3.3 x 0.5 mm)	SLT (Set on <)	3.2 nS	
Functions: ADD, SUB, AND, NOR,		SLT with bypass	3.6 nS	
OR, SLT, BRANCH CONDITION		Quick Compare	1.3 nS	
Shifter	1,848	Data	2.7 nS	0.41W
Functions: SLL, SRL, SRA	(3.4 x 0.5 mm)	Control	2.5 nS	
Integer Multiply and Divide		1 add step	3.7 nS	0.79W
Functions: MULT, MULTU, DIV, DIVU	(4.0 x1.0 mm)			

Table 2: Summary of parameters for major integer processor logic blocks.

throughout the processor. Since DCFL implements multiplexors more naturally than it does the tri-state bus structures more common to CMOS, multiplexors are used to control this flow of operands. A NOR-NOR multiplexor structure is used for its speed with light output loads. Multiplexors are used to select the sources for ALU operands, the desired result from ALU units, and the address for the next instruction. Next instruction possibilities are the next sequential address, a branch address, a jump address, and the address of an exception-handler routine. Nominal delays for the program counter and other major IP logic blocks are given in Table 2.

The controller also initiates the handling of exceptions generated on the IP by updating the registers in CP0 with appropriate information and by directing the program counter to jump to the address of the exception-handling routine. Exceptions such as TLB misses are initiated by the off-chip cache controller. See [15] for a complete description of the MIPS-architecture exception conditions. CP0 provides memory management support, and the system functions, such as state-saving, needed for exception handling. Saving of state in a five stage pipeline requires keeping the four previous program counter values, since an exception can occur in any of the pipe stages. Cache miss exceptions are handled by stalling the pipeline. Because of the static design of datapath circuits, the stall is produced by simply stopping the clock to the datapath. All of CP0 except the TLBs is easily included on the IP chip.

5.2 Register File

The datapath blocks will now be described, beginning with the design of the register file. The 1 K-bit register file is accessed through one write port and two read ports. All ports have parallel data access, providing large data bandwidth. Simultaneous write and read operations to a given register are allowed in the same clock cycle. The RF was conservatively implemented in NOR logic, with super-buffers driving all long lines. An alternative design style, using pass transistors, would result in a smaller memory cell and a faster register file. As noted previously, the MESFET diode gate makes pass-transistor design much more challenging than in insulated gate technologies. The potential speed improvement is not important, since the current RF design meets the target specification, but size is an issue. The current register file limits datapath pitch, and therefore the size of all other datapath units. When the GaAs compiler is developed to the point that it can be used to implement the datapath, a new version of the chip will be designed.

Power and area considerations make the use of super-buffers in the read selector impractical. To improve speed and reduce sensitivity to threshold variation across the RF, a sense amplifier is used for readout. The sense amplifier flip-flop is preset during the first phase of the clock; in the second phase, the selected register bit pulls low one side or the other of the sense-amp flip-flop. The number of long lines required for each cell is minimized by performing all data input and output selection at the memory cell. This has the added benefit of confining the logic to a local area, thereby reducing the number of global logic signals. Some AND logic is used in the memory cell to simplify the logic and reduce area. The AND structure is slower and has reduced noise margin compared to NOR logic, but this was a good trade-off, since the memory cell is in a small area, and most of the delay is in the long interconnect lines. The design has been simulated from 0°C to 70°C with no problems due to threshold shift detected in the AND circuit.

The register file has been simulated using a version of SPICE modified by Vitesse for their E/D MESFET technology. Table 2 shows SPICE predictions of minimum times for a nominal RF chip, with all important layout parasitics included.

5.3 ALU

The ALU performs all of the arithmetic and logic functions required by the MIPS instruction set. Much time was spent optimizing the ALU circuitry and layout, at the polygon level, to meet the 4 nS cycle time specification. This is a case in which higher-level design automation tools could have shortened design time considerably. The worst case instruction is set on less than (SLT). SLT uses the full carry chain plus 2 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. The starting point for the design was the choice of a binary look-ahead tree for carry computation, as described in [16] for CMOS. This design was chosen on the basis that it has low fanout loading, which is important for fast gates in DCFL. Unused gates in the CMOS design were eliminated to save power and parasitic loading; however, this increased the layout design time, since regularity was reduced. First, the critical path was determined and loading on this path was minimized through buffering of secondary paths. Next, gate sizing and buffering along the critical path were determined manually with SPICE simulations. Automated gate sizing and parasitic extraction would have allowed better-informed trade-offs to be made in less time. Automated cell placement and three-layer routing, also being developed under this project, would have reduced time and effort in designing the irregular topologies that resulted from optimizing the ALU for speed.

A conditional inverter is placed on the B input for subtraction, and an extra OR function is performed on the A and B inputs to complete the logic functions not already provided by the carry circuitry. Multiplexors on the output select the actual function performed. For branch condition calculations, "quick compare" circuitry (simple logic comparison without subtracts) and a multiplexor to select the comparison type are added to the ALU. The output of the comparison circuit controls the PC, selecting the actual comparison type, on the instruction following the branch delay slot. A 32 bit wide NOR gate is required to do branch conditional calculations (e.g., branch-on-equal). Such a gate in standard DCFL is too slow to get the result to the PC multiplexor in less than one phase, so a pre-set bus (like the sense amplifier described in the register file) was used, reducing the worst case delay from 3.1 nS to 1.1 nS. The rest of the alu design uses standard DCFL NOR logic with super-buffers on all long lines.

5.4 Shifter

The shifter performs all the shifting operations required by the instruction set. The shifter can perform left and right logical shifts and arithmetic right shifts by 0 to 31 bits. The shifting operation is performed in the ALU time slot; a multiplexor selects between the ALU and shifter outputs. Instead of designing a barrel shifter as commonly used in CMOS, we used a five-stage tree shifter. The first bank of multiplexors converts left shifts to right shifts, and the last bank undoes this at the output. Such an arrangement makes zero insertion and sign extension uniform. The multiplexor design uses a NAND gate with a super-buffer on the output. In a final analysis, this was not the fastest type of multiplexor (see table 3), but it does meet the delay specification with the fewest gates (less power) and with the benefit of the improved voltage swing of super-buffers. This is important for driving lines across 32 bits, as is necessary in the first and last multiplexor banks. However, one must be careful

Mux Type	Buffer	Delay
NOR-NOR	Source Follower	2.0 nS
	Inverter + Super-buffer	2.5 nS
NAND	Super-buffer	2.7 nS
	Inverter	3.8 nS
NOR-NOR	none	4.9 nS
Barrel Shifter	Source Follower	5.6 nS

Table 3: Shifter delay for alternate multiplexor designs.

when driving the upper transistor of a DCFL NAND gate, especially with a superbuffer. If there is no other diode to clamp the line, an incorrect high output can be generated when both inputs are high. While this shifter design is not very area efficient (a large area is needed for interconnect) and has many long lines to drive, it is still faster than a barrel shifter.

5.5 Load Aligner

The load aligner design was more difficult than the shifter because of the timing constraint. In only one phase, the data must be aligned and written into the register file. This is a result of the pipeline design (one instruction branch delay, and two bypass paths) and the assumption that cache accesses will be the critical path. This precludes aligning the data as it comes into the IP (as is done in the CMOS design); instead, it must be aligned in the next pipe stage. The design chosen was similar to the shifter, but the NOR-NOR mux with source follower buffers were used, since it is faster. Here the sign extend is in the critical path since location of the sign bit is not known until the instruction is decoded. This necessitated rearranging the aligner slightly from the straight-forward mux layout, and further optimization to reduce delay.

6 Conclusions

We have examined the motivation and justification for designing with GaAs VLSI for the next generation of fast microprocessors. Many challenges remain in putting together a complete system, but the chips in such a system are within reach of present technology. Performance-driven CAD tools are still needed to aid the design of high speed VLSI. Our experience in designing a GaAs integer processor has illuminated many of the issues involved in GaAs VLSI. Our next goals are completion of the chip set and packaging required for a micro-supercomputer.

7 Acknowledgements

We thank MIPS Computer Systems for supporting this project with technical assistance under a special licensing arrangement. We also gratefully acknowledge the assistance of Seattle Silicon Corp. and Vitesse Semiconductor Corp.

References

[1] R.Van Tuyl, L.Rory, C.A.Liechti, "High speed integrated logic with GaAs MESFET's", *IEEE J. Solid-State Circuits*, Vol.SC-9, 1974, pp.269–276.

[2] W.Jutzi, "Direct coupled circuits with normally-off GaAs MESFETs at 4.3 K," *Arch. Elecktron. und Ubertragungtech, (AEU)*, Vol.25, pp.595-598, 1971.

[3] R.C.Eden, "Comparison of GaAs device approaches for ultrahigh speed VLSI," *Proc. IEEE*, Vol.70 (1), 1982.

[4] B.K.Gilbert, B.A.Naused, D.J.Schwab, and R.L.Thompson, "The Need for a Wholistic Design Approach," Computer, Oct. 1986, pp.29-43.

[5] H.Vlahos and V.Milutinovic, "GaAs Microprocessors and Digital Systems," *IEEE Micro*, Vol.8 (1), 1988, pp.28-56.

[6] R.Zuleeg, J.K.Notthoff, and G.L.Troeger, "Double implanted GaAs complementary JFET's," *IEEE Electron Device Letters* Vol.EDL-5 (1), 1984.

[7] V.Garcia, D.Whitmire, and S.Evans, "A 32b GaAs RISC Microprocessor," ISSCC 1988, San Francisco, CA, Session 2.5.

[8] V.Milutinovic, N.Lopez, and K.Hwang, "A GaAs-Based Microprocessor Architecture for Real-Time Applications," *IEEE Trans. Computers*, June 1987, pp.714-727.

[9] N. Hendrickson, W. Lanrkin, R. Demming, R. Bantolotti, and I. Deyhimy, "A GaAs Bit-Slice Microprocessor Chipset," *IEEE GaAs IC Symposium Tech. Digest*, 1987, pp.197-199.

[10] R.H.Voelker and R.J.Lomax, "Three-Dimensional Simulation of Integrated Circuits Using a Variable Mesh TLM Method," *Microwave and Optical Technology Letters*, vol. 2, no. 4, April, 1989, pp. 125–7.

[11] P.M.Solomon, "A Comparison of Semiconductor Devices for High-Speed Logic," *Proceedings of the IEEE*, Vol.70, No.5, May 1982, p.489.

[12] "10,000-gate Gate Array Challenges ECL Stronghold," Jan. 89, ESD: The Electronic System Design Magazine, p. 19.

[13] J.Rawson, VLSI Technology, Inc., personal communication.

[14] Scott Morrison, Nancy Waller, "Register Windows Vs. General Registers: A Comparison of Memory Access Patterns," University of California, Berkeley, CA, May 13, 1988.

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

[16] N.H.E.Weste and K.Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Co., Reading, Mass., 1985.