An Asynchronous Microprocessor in Gallium Arsenide

José A. Tierno   Alain J. Martin   Drazen Borkovic
Tak Kwan Lee
Department of Computer Science
California Institute of Technology
Pasadena, CA 91125

November 9, 1993

Abstract

In this paper, several techniques for designing asynchronous circuits in Gallium Arsenide are presented. Several new circuits were designed, to implement specific functions necessary to the design of a full microprocessor. A sense-amplifier, a completion tree, and a general circuit structure for operators specified by production rules are introduced. These circuit were used and tested in a variety of designs, including two asynchronous microprocessors and two asynchronous static RAM's. One of the microprocessor runs at over 100 MIPS with a power consumption of 2 Watts.

1 Introduction

With an electron mobility about six times that of silicon at room temperature, and with a lower parasitic capacitance due to semi-insulating substrate, GaAs is potentially faster than silicon. Up until recently, however, GaAs was not available to the VLSI community at large because of inherent fabrication difficulties. These difficulties have been overcome to a large extent. Several foundries are now offering GaAs fabrication lines under conditions similar to CMOS, with density limited to about 100,000 transistors. In particular, Vitesse Semiconductors is offering fabrication through MOSIS to the academic community in the United States.

At the moment, the transistor of choice for GaAs digital VLSI is the MESFET. There is no oxide insulating the gate of a MESFET transistor from source and drain; therefore, the logic families available in GaAs are much less attractive than CMOS or even nMOS. DCFL has been adapted to GaAs, but has very reduced noise margins, and restricted fanin and fanout. With no complementary transistor available, the logic is ratioed. As a result, a considerable fraction of the speed advantage is lost due to the complexity of the available logic families relative to CMOS.

We have developed a design method for asynchronous VLSI that is, to a large extent, technology independent. Thus, it should be straightforward to port a design from one technology to another. Also, since the circuits designed are delay-insensitive, they are more robust with respect to variations in physical parameters. Hence, the method should make it easier to design in a demanding technology such as GaAs, where parameters—particularly threshold voltages—are difficult to control. Finally, since the circuits we design do not use a clock, we avoid the complexities of high-speed clocking schemes. Adapting our method to GaAs design would be an excellent demonstration of the advantages of the method.

From the onset, our intention was to port to GaAs the asynchronous microprocessor we designed in CMOS in 1989 [MBL+89]. At the same time, we would demonstrate the portability of our approach across vastly different technologies, and the efficiency and robustness of the design method.

To this end, several special purpose circuits had to be designed. Some of them are presented in this paper, together with their use in the microprocessor.

2 GaAs and the MESFET

2.1 MEtal Semiconductor Field Effect Transistor

The MEtal Semiconductor Field Effect Transistor, or MESFET, is the transistor of choice for GaAs VLSI applications. It is the easiest to manufacture, provides the highest density (about 100,000 transistors on a chip), and has a very good power-delay product, that competes fairly well with CMOS. MESFET circuits can be better compared to ECL. At this point, GaAs is slightly faster, uses far less power, and has higher circuit density, for a similar cost. An important application of GaAs is to provide replacements for ECL parts, specially fast RAM's, and other LSI circuits.

MESFET's are junction FET's. The gate is built with metal, and forms a Schottky junction with the transistor channel. There is no insulation between gate and channel; the Schottky junction creates a diode from gate to source and from gate to drain. This diode is probably the most serious constraint for MESFET circuits, since it limits the voltage differential between gate and source or drain to the diode's forward conduction voltage, about 0.7 volts. Above that difference, the diode will be forward biased and current will flow from the gate into the channel.

Hole mobility is low in GaAs (10 times less than electron mobility [LB90]), which makes p-type FETs fairly slow. There is, therefore, no complementary logic available. As in nMOS, n-type transistors come in two flavors, depletion-

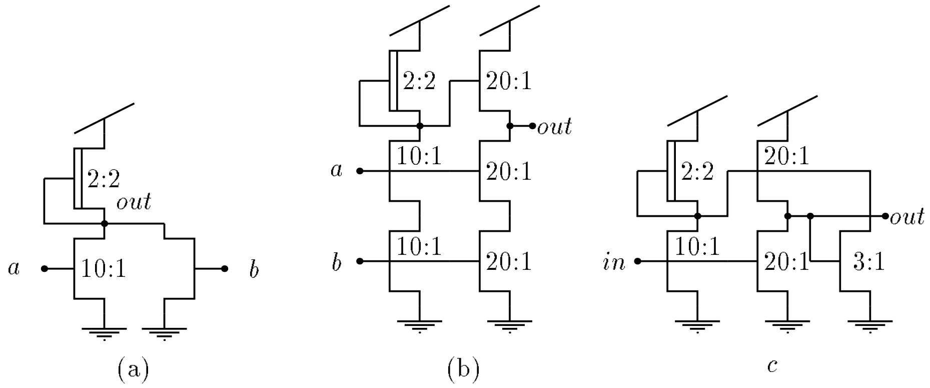


Figure 1: (a) DCFL nor gate, (b) super buffered nand gate, and (c)"squeeze" buffer

mode and enhancement-mode. E-mode transistors have a positive threshold voltage, and d-mode have a negative threshold voltage, that is, they require a negative gate to source voltage to be cut off.

2.2 Direct Coupled Fet Logic

Analogous to its nMOS namesake, DCFL is the most widely used logic family in GaAs VLSI. It is simple, uses little power, and has the highest density of all. Figure 1(a) shows a DCFL nor gate.

Signals have a restricted voltage swing, because of the input-to-ground diode at the input of DCFL gates. Logic-low is about 0.1V, while logic-high is about 0.6V; this drastically reduces the noise margins of DCFL gates. In DCFL nand-gates noise margins become critical: the series transistors in the pull-down chain push the logic low to about 0.2V, and the noise margin is so small that many designers avoid using nand gates completely.

As in nMOS, signals often have to be buffered. A super-buffer configuration can be used (see Figure 1(b) and (c)). A super-buffer also increases the noise margins by lowering the logic-low voltage, since the output stage is not ratioed.

3 Description of the Processor

The microprocessor is a 16-bit, pipelined, RISC-style processor. It is a modified version of the CMOS design described in [MBL$^+$89].

Instructions are issued in order, but may complete out of order. The processor has 16 general purpose registers, with four buses, two for read and two for write. Registers have individual locks to solve read-after-write and write-after-write hazards.

In addition to the ALU, there is one adder in the program counter (PC) unit, for relative branching and incrementing the program counter register. For simplicity, the memory address adder was omitted, thus reducing the number of required buses from five to four, and reducing the size of the datapath compared to the CMOS processor.

Other modifications include a revised pipelining of the ALU unit, and better balanced control of some of the shared resources.

The processor was initially specified as a set of concurrent processes. The text of these processes, shown in Figures 2 and 3 was later transformed into a signal transition language, or handshaking expansion, and then compiled into the gate netlist of the final circuit.

The high-level specification of Figures 2 and 3 shows in detail how the different units interact. The language used is similar to C.A.R. Hoare's CSP [Hoa78], and described in [Mar90].

4 GaAs Technology Mapping

Even though it is be tempting to map the processor design into DCFL gates, the specific requirements of asynchronous circuits makes that choice impractical.

A robust DCFL circuit has to be made almost exclusively from nor gates, plus super-buffers and a few special purpose circuits. We could build the processor with these elements, but at the expense of compromising the delay-insensitivity of the circuit in almost every gate. It is usually safe to implement C-elements with nor gates; however, in such a large design it would be almost impossible to keep track of delays and make sure that no hazard threatens the functionality of the circuit. Parts of the layout are generated with standard-cell-place-and-route tools, where we do not have as much control over delays as we need.

The complexity and number of inputs that a DCFL gate can have are limited by noise margins, subthreshold currents, and variations in threshold voltages. The synthesis method sometimes generates large or complex operators, and these would have to be decomposed into smaller ones to fit in the available smaller DCFL gates. This decomposition creates a number of extra nodes that may introduce hazards or races. A proper decomposition of the bigger operators is, at best, a very hard task. We need circuits that allow a direct synthesis of those operators, up to a reasonable size. To solve the problem of large operators, we have investigated several alternative logic configurations.

$$
\begin{array}{l}
IMEM \equiv *[ID!imem[pc]]
\end{array}
$$

$$
\begin{array}{l}
FETCH \equiv *[PCI1;\ ID?i;\ PCI2;\ E1!i; \\
\qquad\qquad [\ \ \ offset(i.op) \longrightarrow\ PCI1;\ ID?offset;\ PCI2;\ OF \\
\qquad\qquad [\!]\ \neg offset(i.po) \longrightarrow\ \mathbf{skip} \\
\qquad\qquad];\ E2 \\
\qquad\quad]
\end{array}
$$

$$
\begin{array}{l}
PCADD \equiv (\quad *[[\ \overline{PCI1}\ \longrightarrow\ PCI1;\ y := pc + 1;\ PCI2;\ pc := y \\
\qquad\qquad\qquad [\!]\ \overline{PCA1} \longrightarrow\ PCA1;\ y := pc + offset;\ PCA2;\ pc := y \\
\qquad\qquad\qquad [\!]\ \overline{Xpc}\ \ \ \longrightarrow\ X!pc \bullet Xpc \\
\qquad\qquad\qquad [\!]\ \overline{Ypc}\ \ \ \longrightarrow\ Y?pc \bullet Ypc \\
\qquad\qquad\quad]] \\
\qquad\qquad \|\ \ *[[\ \overline{Xof} \longrightarrow\ X!offset \bullet Xof\ \ ]] \\
\qquad\qquad)
\end{array}
$$

$$
\begin{array}{l}
EXEC \equiv *[E1?j; \\
\qquad\qquad [\ alu(j.op)\ \ \longrightarrow\ E2;\ Xs \bullet Ys \bullet AC!j.op \bullet ZAs \bullet P \\
\qquad\qquad [\!]\ ld(j.op)\ \ \ \longrightarrow\ E2;\ ZMs \bullet Ys \bullet MCL \\
\qquad\qquad [\!]\ st(j.op)\ \ \ \longrightarrow\ E2;\ Xs \bullet Ys \bullet MCS \\
\qquad\qquad [\!]\ adi(j.op)\ \ \longrightarrow\ OF;\ E2;\ Xof \bullet Ys \bullet AC!add \bullet ZAs \bullet P \\
\qquad\qquad [\!]\ stpc(j.op) \longrightarrow\ Xpc \bullet Ys \bullet AC!add \bullet ZAs \bullet P;\ E2 \\
\qquad\qquad [\!]\ jmp(j.op) \longrightarrow\ Ypc \bullet Ys;\ E2 \\
\qquad\qquad [\!]\ brch(j.op) \longrightarrow\ OF;\ F?f; \\
\qquad\qquad\qquad\qquad\qquad\quad [\ \ \ cond(f, j.cc) \longrightarrow\ PCA1;\ PCA2 \\
\qquad\qquad\qquad\qquad\qquad\quad [\!]\ \neg cond(f, j.cc) \longrightarrow\ \mathbf{skip} \\
\qquad\qquad\qquad\qquad\qquad\quad];\ E2 \\
\qquad\quad]]
\end{array}
$$

Figure 2: The program, first part

$$
\begin{array}{lllll}
ALU \equiv (& *[[\ \overline{AC} & \longrightarrow & AC?op \bullet X?x \bullet Y?y; & \\
 & & & \langle z, f\rangle := aluf(x, y, op, f) \bullet B & \\
 & \quad \square\ \overline{F} & \longrightarrow & F!f & \\
 & \quad]] & & & \\
 \| & *[B;\ ZA!z \bullet V] & & & \\
 \| & *[P;\ V] & & & \\
) & & & &
\end{array}
$$

$$
\begin{array}{llll}
MU \equiv *[[& \overline{MCL} & \longrightarrow & Y?ma \bullet MCL;\ \ MDL?w;\ \ ZM!w \\
\quad \square & \overline{MCS} & \longrightarrow & X?w \bullet Y?ma \bullet MCS;\ \ MDS!w \\
]] & & &
\end{array}
$$

$$
\begin{array}{llll}
DMEM \equiv *[[& \overline{MDL} & \longrightarrow & MDL!dmem[ma] \\
\quad \square & \overline{MDS} & \longrightarrow & MDS?dmem[ma] \\
]] & & &
\end{array}
$$

$$
\begin{array}{llll}
REG[k] \equiv (& *[[\ \neg bk \wedge k = j.x \wedge \overline{Xs} & \longrightarrow & X!r \bullet Xs\ ]] \\
\| & *[[\ \neg bk \wedge k = j.y \wedge \overline{Ys} & \longrightarrow & Y!r \bullet Ys\ ]] \\
\| & *[[\ \neg bk \wedge k = j.z \wedge \overline{ZAs} & \longrightarrow & bk\uparrow;\ \ ZAs;\ \ ZA?r;\ \ bk\downarrow\ ]] \\
\| & *[[\ \neg bk \wedge k = j.z \wedge \overline{ZMs} & \longrightarrow & bk\uparrow;\ \ ZMs;\ \ ZM?r;\ \ bk\downarrow\ ]] \\
) & & &
\end{array}
$$

Figure 3: The program, second part

4.1 Datapath

The datapath comprises three different units: the ALU, the PC unit for manipulations of the program counter, and the memory unit for execution of load and store operations. They are mostly combinatorial circuits, replicated a number of times. Data-path delay is determined mostly by carry-chain, control signal, and bus delays. Carry-chain delay is data dependent, since the adder uses carry prediction [Mar92].

It is important that the datapath be optimized for size: in the datapath most signals are local, with the exception of control lines and buses, and thus delays and power depend directly on the physical dimensions of the datapath.

The datapath must be optimized for power; in this design, for example 70% of all power is spent by the datapath, 15% by the register file, and 15% by the control logic (not counting pad-driver power).

To satisfy these constraints, all datapath gates are DCFL, except nand gates and buffers, and completion-detection circuits.

Nand gates were implemented as shown in Figure 1(b) [LB90]. The super-

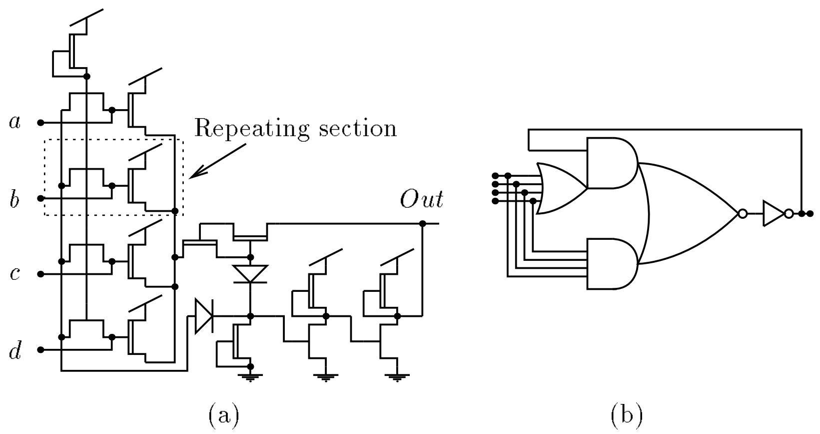


Figure 4: Multi-input C-element. (a) Transistor schematic, and (b) logical diagram

buffer stage allows the output low voltage to be low enough, since the pull-down does not have to fight a passive pull-up. The lower output low increases the noise margins considerably, with a small penalty in area and power.

Super-buffers were used to buffer bus and control signals. To improve performance and noise margin characteristics, a feed-back transistor was added, creating a "squeeze" buffer (see Figure 1(c) [Bro92]). Squeeze buffers allow the use of a stronger pull-up transistor, the feed-back transistor limits the output high voltage.

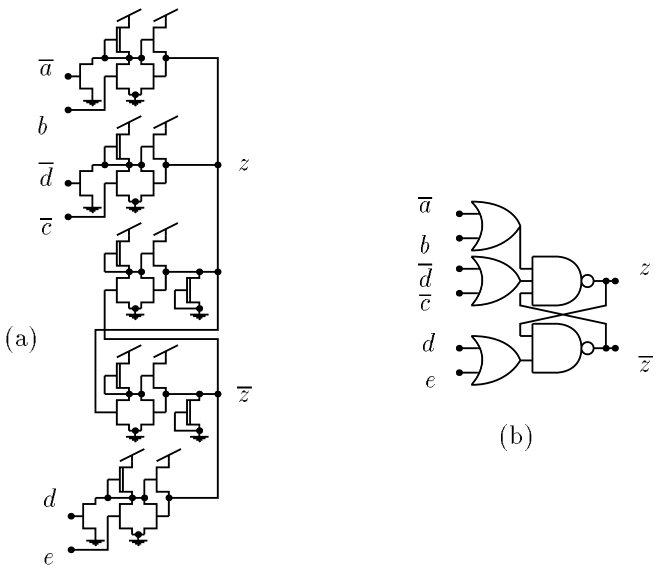
To generate completion signals from the datapath, we use C-elements with a large number of inputs. They can be built from smaller C-elements connected in a tree [Mar90], or, as in this case, as a single logic gate (see Figure 4(a) and (b)). For a discussion on how this circuit works, see [Tie92]. Though a completion tree could be implemented with DCFL nor gates, it would be significantly slower and bigger than that of Figure 4(a). Completion detection is in sequence with the calculations performed by the datapath, and affects performance directly. It is critical to have an easy and fast way of generating the completion signal.

4.2 Control Logic

Control logic takes care of the sequencing of actions in the processor. After compilation, each control signal is assigned a set of "production rules", of the form:

$$\begin{array}{lcl} G & \rightarrow & z\uparrow \\ H & \rightarrow & z\downarrow \end{array}$$

where G and H are boolean expressions in terms of the other signals. G and H do not have to be complementary. In fact, most operators in the control are



$$a \wedge \neg b \vee d \wedge c \rightarrow z \uparrow$$
$$\neg d \wedge \neg e \rightarrow z \downarrow$$

(c)

Figure 5: Dual rail implementation of control signals. (a) Transistor schematic, (b) logical diagram, and (c) production rules

state holding, that is, $G \vee H$ does not hold. There are different ways of giving a direct implementation of these production rules. One, Source Follower Fet Logic, is given in [Tie92]. In that paper, a systematic way of generating any operator described by production rules is presented. This method was applied in the design of the first GaAs microprocessor. However, it resulted in a circuit with a large power consumption (4 Watts) and modest performance (70 MIPS).

For the second GaAs processor, a different approach was used. Each signal is implemented as a dual-rail encoded signal, always generating both positive and negative sense. Since most signals are state-holding, this is necessary anyway (we do not have dynamic logic available). On the up side, no inverters are necessary to generate the boolean expressions for the production rules. On the down side, combinatorial gates require extra circuitry to generate dual-rail outputs, as do signals coming in from the datapath.

Figure 5 shows how a specific set of production rules are implemented in dual-rail. Note the feed-back transistor on the outputs of the individual nor gates. These have the same function as the one on the "squeeze" buffer, allowing us to use much stronger pull-up transistors.

4.3 PLA's

The power consumption of these circuits is relatively high, and the chips run hot (around $100°C$). At this temperature, subthreshold currents of the pull-down

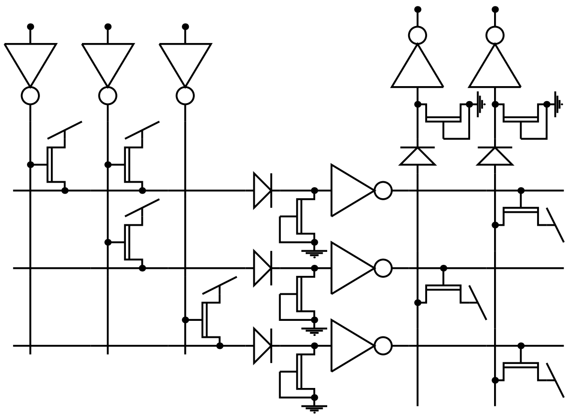


Figure 6: Example of a nor-nor PLA implemented with source-followers

transistors may be strong enough to overpower the pull-ups; nor gates with more than 6 inputs are impractical, because the off current of 6 transistors is of the same order of magnitude as the on current of one transistor. Therefore, static DCFL PLAs cannot be used.

We use a different structure for the PLAs in the processors. The nor planes are implemented with source followers, which can be turned off more effectively than the corresponding DCFL structure (see Figure 6). A penalty is paid in speed and power, but min-terms with up to 10 inputs are realizable. The internal signals in the nor plane can switch rail to rail, giving much improved noise margins, and the level shifting diodes help reverse-bias the pull-up transistors of the source-followers. Also, the ratio between the pull-up and pull-down in the source-follower is close to 1, and the subthreshold current of the pull-down can better balance the pull-up.

4.4 Register File

The register file has 16 registers, 16 bits wide, and four ports, two for read and two for write. Register 0 is hard-wired to be read always as zero. Each register bit has a total of 12 transistors, four for the flip-flop and two extra for each port (see Figure 7). All ports are dual-rail, that is, data and inverted data is provided. Between reads and writes the buses are pre-charged to a neutral value, to prepare for the next operation, and reset the completion circuits.

Read ports are implemented with a dual-ended sense amplifier (see Figure 8). This sense-amp detects a small difference between the true and false buses, and drives these buses strongly in opposite directions, using transistors T5, T4, and T1. To work properly, the register that is being read has to be selected some time prior to applying the sense signal. To this effect, the sense signal is derived from an "or" of all select signals for the given port.

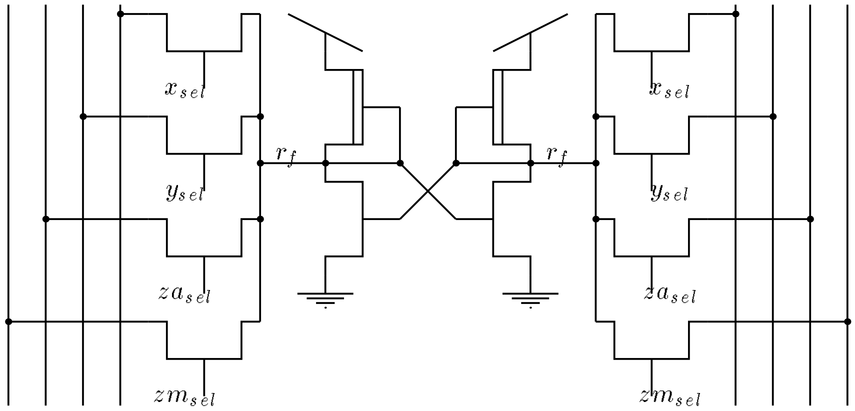


Figure 7: Register cell

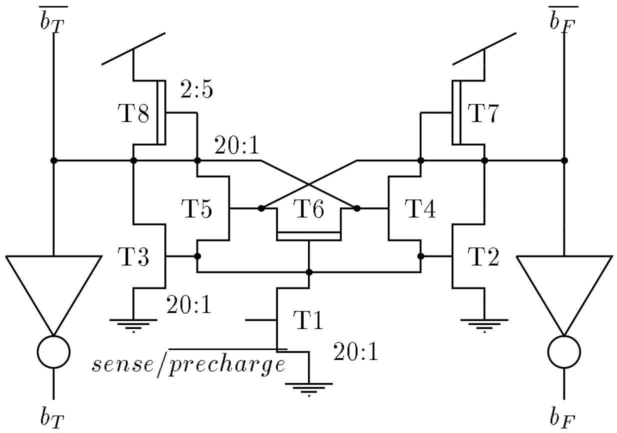


Figure 8: Sense amplifier, and pre-charge circuit for the register file

When sense is off, transistors T8,T3 and T7,T2 pre-charge buses $\overline{b_T}$ and $\overline{b_F}$ to a value determined by the ratio between T8 and T3 (T7 and T2). Transistor T6 further ensures the symmetry of the circuit.

The buses are then buffered before going into the datapath, to isolate the sense amplifier, and to restore the logic levels of the data signals.

5 Results

Using these techniques, several circuits have been designed, fabricated and tested on the HGAAS II and HGAAS III processes offered by Vitesse Semiconductors. Among them, small RAM's, register files, and two microprocessors.

5.1 Asynchronous Static Memories

We designed and fabricated two different types of static memories in GaAs. The first is a dual ported register file, 16 words of 4 bits/word. It was meant to provide a small amount of fast memory for the microprocessor to run test programs at full speed. Out of 30 bonded devices, 29 were functional. Access time is 5ns, and the chip dissipates 500mW at 2.2V.

The second SRAM has 64 words of 4bits/word [Hof91], and was designed as an intermediate step towards a larger memory to be used as a cache for the processor. All 30 bonded devices received were functional. The access time is 3ns, including pad delays, and the chip dissipates 700mW at 2.3V

This 64×4 memory was designed after the first processor, and incorporates several improvements derived from our experience with the earlier designs. Also, the circuits were carefully optimized for high speed and low power consumption. The performance obtained indicates that the improvements envisioned for the next processor generation should be attainable.

5.2 Microprocessors

The first microprocessor design uses the circuits described in [Tie92]. The main concern during the design was to get around parameter variation problems and noise margin considerations. This was achieved, but at the expense of power and performance. At 70 MIPS, the processor consumes 4 Watt.

Several new ideas and circuits were incorporated in the new design, many of them presented in this paper. All of those circuits were fabricated and tested successfully.

This design was extensively simulated with Hspice. The expected performance of this design is about 200 MIPS with a dissipation of 2 Watts, fabricated with the HGAAS III process. Power and speed predictions have been very accurate so far using the Hspice models. However, in this case the measured performance was only 100 MIPS. The causes are still under investigation. There is some evidence of fabrication problems, and underestimation of the parasite capacitances as extracted by the MAGIC layout program.

6 Conclusions

In the course of this project, we have designed a number of very different GaAs circuits. Most of them had very strict requirements, to overcome the limitations of GaAs. No general solution is given to synthesize all logic circuits in a design as big as a microprocessor. Instead, we found specific solutions to implement completion trees, control circuits, PLA's, registers, etc.

The first GaAs microprocessor, though disappointing in terms of performance, gave us invaluable experience in verifying and testing these circuits, as well as which were the inherent problems of each. Together with some of the RAM designs, it allowed us to prepare for a second microprocessor, with considerable improvements in performance.

Another factor affecting performance is pad delay. So far we have used ECL levels on the outside, to be able to interface to standard parts and simplify prototyping. Pad delays are in the order of 1ns, mostly spent in level conversion; also, the padframe uses a considerable amount of power —close to $1A$ worst case for the processor—. This delay and power can be greatly reduced by designing matched pad drivers and receivers in a system composed exclusively of GaAs parts. It would certainly be a requirement in the interface with cache memory.

Porting the design of the original CMOS microprocessor was almost as easy as expected. A few changes were necessary because of the complexity of register cells in the first GaAs version. These changes were carried over to the second processor to speed-up the redesign.

All considered, the expected performance of the new microprocessor is satisfactory. At 50 MIPS/watt, it offers remarkable speed for the power consumption.

Acknowledgments

We are indebted to Marcel van der Goot for his help in generating high quality software support. Acknowledgment is due to Steve Burns and Pieter Hazewindus, for their participation in the original design, and many very useful comments and conversations, and H. Peter Hofstee for his collaboration in the design of asynchronous memories. Many thanks to Ray Milano of Vitesse Semiconductors, for invaluable help with the HGAAS technology, and to Cindy Hibbert and Metasoftware, for their help with Hspice.

The research described in this paper was sponsored by the Defense Advanced Research Projects Agency, DARPA Order number 6202, and monitored by the Office of Naval Research under contract number N00014-87-K-0745.

References

[Bro92] Richard Brown. Private communication, 1992.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Comm. ACM*, 21(8):666–677, 1978.

[Hof91] H. P. Hofstee. Deriving some asynchronous memories. Unpublished, 1991.

[LB90] S. I. Long and S. E. Butner. *Gallium Arsenide Digital Integrated Circuit Design*. McGraw-Hill, New York, 1990.

[Mar90] Alain J. Martin. Synthesis of asynchronous VLSI circuits. In J. Straunstrup, editor, *Formal Methods for VLSI Design*, pages 237–283. North-Holland, 1990.

[Mar92] A. J. Martin. Asynchronous datapaths and the design of an asynchronous adder. *Formal Methods in System Design*, 1(1):117–137, 1992.

[MBL+89] Alain J. Martin, Steven M. Burns, T. K. Lee, Dražen Borković, and Pieter J. Hazewindus. The design of an asynchronous microprocessor. In Charles L. Seitz, editor, *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pages 351–373. MIT Press, 1989.

[Tie92] J. A. Tierno. Designing asynchronous circuits in Gallium Arsenide. CS-TR-92-19, California Institute of Technology, 1992.