A 100-MIPS GaAs Asynchronous Microprocessor

JOSÉ A. TIERNO
ALAIN J. MARTIN
DRAZEN BORKOVIC
TAK KWAN LEE
California Institute of Technology

The authors describe how they ported an asynchronous microprocessor previously implemented in CMOS to gallium arsenide, using a technology-independent asynchronous design technique. They introduce new circuits including a sense-amplifier, a completion detection circuit, and a general circuit structure for operators specified by production rules. The authors used and tested these circuits in a variety of designs.

WE HAVE DEVELOPED a design method for asynchronous VLSI circuits that is, to a large extent, technology independent.[1] Thus, it makes porting a design from one technology to another straightforward. Also, since the circuits designed by this method are quasi-delay-insensitive, they are more robust with respect to variations in physical parameters. Hence, the method facilitates designing in a demanding technology such as GaAs (gallium arsenide), in which fabrication process parameters, particularly threshold voltages, are difficult to control. Finally, since asynchronous circuits do not use a clock, we avoid the complexities of high-speed clocking schemes.

Adapting our method to GaAs design is an excellent demonstration of the method's advantages. Thus, we decided to port the asynchronous microprocessor we designed in CMOS in 1989 to GaAs.[2] We would demonstrate the method's portability across vastly different technologies, as well as its efficiency and robustness.

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. Until recently, however, GaAs was not available to the VLSI community at large because of inherent fabrication difficulties. These difficulties have largely been overcome. Several foundries now offer GaAs fabrication lines under conditions similar to CMOS fabrication, with chip size limited to about 100,000 transistors. In particular, Vitesse Semiconductors offers fabrication through MOSIS (Metal Oxide Semiconductor Implementation System) to the United States academic community.

Currently, the transistor of choice for GaAs digital VLSI circuits is the MESFET (metal semiconductor field-effect transistor). With no oxide to insulate a MESFET gate from source and drain, the logic families available in GaAs are much less attractive than in CMOS or even NMOS. DCFL (direct coupled field-effect transistor logic) has been adapted to GaAs, but it has greatly reduced noise margins and restricted fan-in and fan-out. With no complementary transistor available, the logic is ratioed. As a result, GaAs loses a considerable fraction of its speed advantage due to the complexity of the available logic families. To bypass these limitations, we designed several new circuits and adapted a number of old circuits for use in the microprocessor.

0740-7475/94/$04.00 © 1994 IEEE

```
IMEM ≡ *[ ID!imem[pc] ]

FETCH ≡ *  [ PCI1; ID?i; PCI2; E1!i;
             [    offset(i.op)  → PCI1; ID?offset; PCI2; OF
             [] ¬ offset(i.op)  → skip
             ];   E2
           ]

PCADD ≡ (   *[ [ PCI1  → PCI1; y := pc + 1; PCI2; pc := y
             [] PCA1  → PCA1; y := pc + offset; PCA2; pc := y
             [] Xpc   → X!pc • Xpc
             [] Ypc   → Y?pc • Ypc
             ] ]
          || *[ [ Xof   → X!offset • Xof ] ]
          )

EXEC ≡ *[E1?j;
          [ alu(j.op)  →  E2; Xs • Ys • AC!j.op • ZAs • P
          [] ld(j.op)  →  E2; ZMs • Ys • MCL
          [] st(j.op)  →  E2; Xs • Ys • MCS
          [] adi(j.op) →  OF; E2; Xof • Ys • AC!add • ZAs • P
          [] stpc(j.op)→  Xpc • Ys• AC!add • ZAs • P; E2
          [] jmp(j.op) →  Ypc • Ys; E2
          [] brch(j.op)→  OF; F?f;
                          [    cond(f,j.cc) → PCA1; PCA2
                          [] ¬ cond(f,j.cc) → skip
                          ];   E2
          ]]
```

```
ALU ≡ ( * [ [    AC → AC?op • X?x • Y?y;
                      ⟨z,f⟩ := aluf(x,y,op,f) • B
           [] F  → F!f
           ] ]
      || • [B; ZA!z • V]
      || * [P; V]
      )

MU ≡ *  [ [ MCL → Y?ma • MCL; MDL?w; ZM!w
          [] MCS → X?w • Y?ma • MCS; MDS!w
          ] ]

DMEM ≡ * [ [ MDL → MDL!dmem [ma]
           [] MDS → MDS?dmem [ma]
           ] ]

REG[k] ≡ ( *[ [ ¬bk ∧ k = j.x ∧ Xs   → X!r •  Xs ] ]
         || *[ [ ¬bk ∧ k = j.y ∧ Ys   → Y!r •  Ys ] ]
         || *[ [ ¬bk ∧ k = j.z ∧ ZAs  → bk↑;  ZAs; ZA?r; bk↓ ] ]
         || *[ [ ¬bk ∧ k = j.z ∧ ZMs  → bk↑;  ZMs; ZM?r; bk↓ ] ]
         )
```

Figure 1. *CHP description of the microprocessor.*

The microprocessor

The microprocessor, a 16-bit, pipelined RISC (reduced instruction-set computer), is a modified version of the 1989 CMOS design. Instructions issue in order but may complete out of order. The microprocessor 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 conflicts.

In addition to the ALU, the program counter (PC) unit contains one adder for relative branching and incrementing the PC register. For simplicity, we omitted the CMOS microprocessor's memory address adder, thus reducing the number of required buses from five to four and making the data path smaller. Other modifications include a revised pipelining of the ALU unit and a revised sequencing circuit to equalize delays in all pipeline stages.

We initially specify the microprocessor as a set of concurrent processes. We later transform the text of these processes, shown in Figure 1, into a signal transition language, or handshaking expansion, and then we compile it into production rules (the gate netlist) for the circuit.

The high-level specification of Figure 1 shows in detail how the different units interact. (The language used, Communicating Hardware Processes, or CHP, is similar to Hoare's Communicating Sequential Processes, or CSP.[3,4]) Process FETCH fetches instructions from the instruction memory and transmits them to process EXEC, which decodes them. Process PCADD updates the address of the next instruction concurrently with the instruction fetch and controls the offset register. The execution of an ALU instruction by process ALU can overlap the execution of a memory instruction by process MU. EXEC executes the jump and branch instructions. The ALU executes stpc as the instruction "add the contents of register pc to register y and store it in register z." The process array REG[k] implements the register file, each register having a lock. PCADD contains its own adder. Processes IMEM and DMEM describe the instruction memory and the data memory, respectively.

These concurrent processes synchronize by means of communication commands on channels. A restricted form of shared variables is allowed. Figure 2 shows the structure of processes and channels.

GaAs and the MESFET

The MESFET is the best-suited transistor for GaAs VLSI applications. It is the easiest to manufacture, provides the highest density (about 100,000 transistors on a chip), and, for clock frequencies of more than 200 MHz, has a better power-delay product than CMOS. Compared with ECL (emitter-coupled logic), GaAs is slightly faster, uses far less power, and has higher circuit density, for a similar cost. An important application of GaAs is to replace ECL parts, such as fast RAMs, and other LSI circuits.

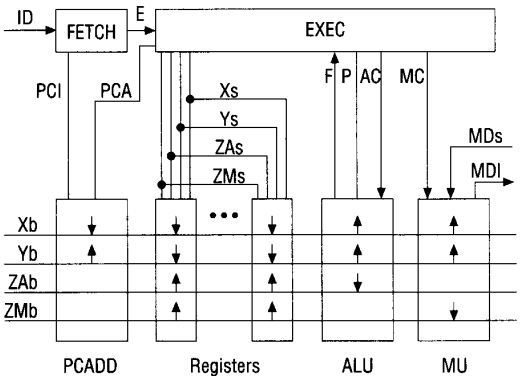


Figure 2. *Process and channel structure.*

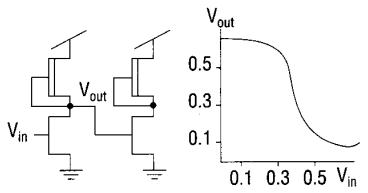


Figure 3. *Input-output characteristic of a DCFL inverter driving a DCFL inverter (simulated).*

MESFETs are junction FETs. The metal gate forms a Schottky junction with the transistor channel. With no insulation between gate and channel, the Schottky junction creates a diode from gate to source and from gate to drain. This diode imposes severe limitations on the type of gates that digital circuits can employ. At around 0.6V voltage differential between gate and drain (or gate and source), the gate-to-drain (or gate-to-source) diode starts conducting a significant amount of current. In a configuration like a DCFL inverter, if the input voltage goes higher than 0.7V, the gate-to-drain voltage (that is, the gate-to-output voltage) becomes larger than 0.6V. Then current can flow from the input to the output, causing the characteristic DCFL transfer curve shown in Figure 3. Observe that the output voltage starts to increase once the input voltage goes beyond 0.6V.

Hole mobility is low in GaAs (10 times less than electron mobility[5]), making p-type FETs too slow relative to n-type. Therefore, complementary logic is not practical. As in NMOS, n-type transistors come in two flavors: enhancement mode and depletion mode. E-mode transistors have a positive threshold voltage. D-mode transistors have a negative threshold voltage; that is, they require a negative gate-to-source voltage to be cut off.

Direct coupled FET logic. DCFL, the most widely used logic family in GaAs VLSI, is analogous to its NMOS counterpart. It is simple, uses little power, and has the highest density of all. Figure 4a 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 gate-to-source voltage in the top transistor of the pull-down chain is even closer to the transistor's threshold voltage, making the noise margin so small that many designers avoid using NAND gates altogether.

As in NMOS, signals often must be buffered. A superbuffer configuration, shown in Figures 4b and 4c, increases the noise margins by lowering the logic-low voltage, since the output stage is not ratioed.

Power consumption. In DCFL GaAs, most of the power dissipation comes from static currents. DCFL gates take current continuously, with only a small part used to charge and discharge capacitors. We obtain a good power-delay product in GaAs because transistors are faster and gate capacitances are lower than in CMOS, and the reduced delay makes up for the extra power.

To optimize the power-delay product, we design circuits with little redundancy, so that most of the circuit is active at any time. For example, we prefer a pipelined ALU to two nonpipe-

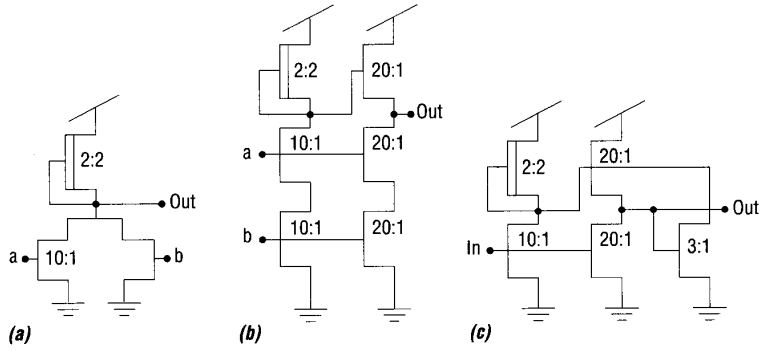


Figure 4. *DCFL NOR gate (a); superbuffered NAND gate (b); squeeze buffer (c).*

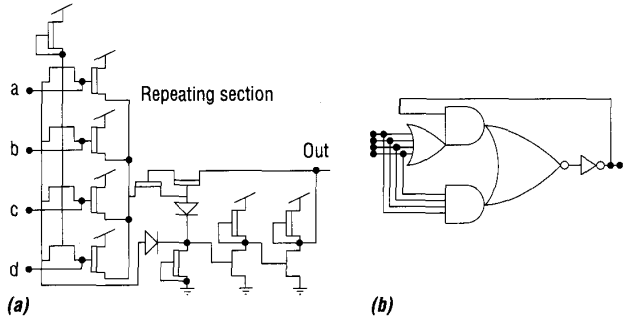


Figure 5. *Multi-input C-element: transistor schematic* ***(a)****; logical diagram* ***(b)****.*

lined ALUs in parallel because the energy cost of the first is half that of the second, virtually independent of usage.

It is possible to build dynamic GaAs circuits with very low power consumption.[6,7] However, the practical circuits demonstrated so far have very low integration levels.

GaAs technology mapping

Although mapping the microprocessor design into DCFL gates is tempting, the specific requirements of asynchronous circuits make that choice impractical. DCFL has low noise margins, and the gates one can use in practice are reduced to NOR gates with a relatively small number of inputs. Asynchronous operation requires monotone signal transitions and is sensitive to noise and charge sharing. Also, the compilation process sometimes generates complex gates with a large number of inputs. Delay-insensitive decomposition of these gates into smaller gates is a delicate task.

We need circuits that allow a direct synthesis of complex gates up to a reasonable size. To solve the problem of large operators, we have investigated several alternative logic configurations. We chose different solutions for the data path and the control circuits.

Data path. The data path includes 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. These circuits consist of combinatorial cells with no feedback loops, replicated a number of times. Data path delay is determined primarily by carry-chain, control signal, and bus delays.[8] With no feedback loops to amplify noise, noise problems are less severe.

We must optimize the data path for size and power. Most signals in the data path are local, with the exception of control lines and buses, and delays and power depend directly on the data path's physical dimensions. In this design, 70% of all power is spent in the data path, 15% in the register file, and 15% in the control logic (pad driver power is excluded from this computation).

To satisfy size and power constraints, all data path gates are DCFL, except NAND gates, buffers, and completion detection circuits. DCFL gates are small and power efficient. The data path contains only the simple gates available in DCFL.

Figure 4b shows the implementation of NAND gates.[5] The superbuffer stage allows the output low voltage to be low enough for other DCFL gates because the pull-down does not have to fight a passive pull-up. Therefore, noise margins increase considerably, with a small penalty in area and power.

We use superbuffers to buffer bus and control signals. To improve performance and noise margin characteristics, we added a feedback transistor, creating a squeeze buffer (see Figure 4c). Squeeze buffers (developed by Richard B. Brown of the University of Michigan) allow the use of a stronger pull-up transistor; the feedback transistor limits the output high voltage.

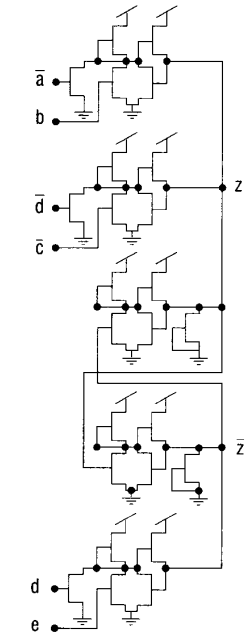
Completion detection takes place in sequence with the calculations performed by the data path and affects performance directly. Generating the completion signal efficiently is critical. To generate completion signals from the data path, we use C-elements with a large number of inputs. They can be built from smaller C-elements connected in a tree,[4] or, as in our microprocessor, as a single logic gate (see Figure 5).[9] Though we could implement a completion tree with DCFL NOR gates, it would be significantly slower and bigger than that of Figure 5a.

Control logic. Control logic takes care of the sequencing of actions in the microprocessor. Compilation assigns each control signal a set of production rules of the form

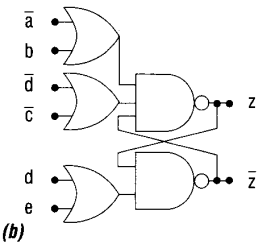
$$G \rightarrow z\uparrow$$
$$H \rightarrow z\downarrow$$

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 state holding; that is, $G \vee H$ does not hold. There are different direct implementations of these production rules. One, called source follower FET logic, is described in Tierno's paper, which presents a systematic way of generating any operator described by production rules.[9] We applied this method in the design of our first GaAs microprocessor. However, it resulted in a circuit with a large power consumption (4W) and modest performance (70 MIPS).

For our second GaAs microprocessor, we used a different approach. Each



(a)



(b)

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

(c)

Figure 6. *Dual-rail implementation of control signals: transistor schematic* ***(a)****; logical diagram* ***(b)****; production rules* ***(c)****.*

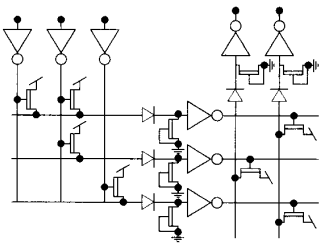


Figure 7. *Example of a NOR-NOR PLA implemented with source followers.*

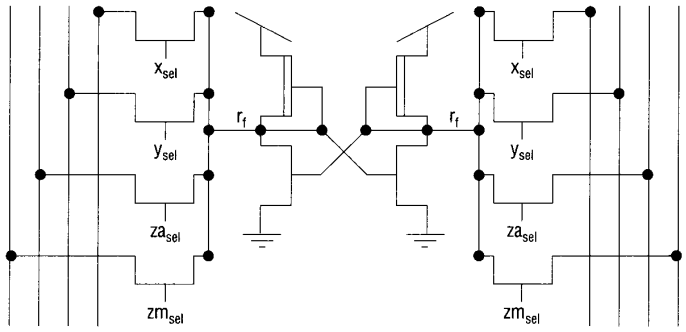


Figure 8. *Register cell.*

signal has a dual-rail encoding, always generating both a positive and a negative sense. An advantage of this approach is that no inverters are necessary to generate the Boolean expressions for the production rules. A drawback is that combinational gates require extra circuitry to generate dual-rail outputs, as do signals coming in from the data path.

Figure 6 shows how a specific set of production rules is implemented in dual-rail. Note the feedback transistor on the outputs of the individual NOR gates. These transistors have the same function as the one on the squeeze buffer, allowing the use of much stronger pull-up transistors.

Programmable logic arrays. The microprocessor uses PLAs to evaluate condition codes for the branch instructions and kill-propagate-generate codes for the ALU. The microprocessor's power consumption is relatively high (2W), and the chips run hot (around 100°C). At this temperature, subthreshold currents of the pull-down transistors may be strong enough to overpower the pull-ups. NOR gates with more than six inputs are impractical because the off current of six transistors is of the same order of magnitude as the on current of one transistor. Therefore, we cannot use static DCFL PLAs.

We implemented the NOR planes with source followers, which can be turned off more effectively than the corresponding DCFL structure (see Figure 7). We pay a penalty 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; the diodes act as level-shifters and help cut the input transistors. Also, the width ratio between the pull-up and pull-down transistors in the source follower is close to 1, and the subthreshold current of the pull-down can better balance the pull-up.

Register file. The register file has 16 registers, each 16 bits wide, and four ports—two for reading and two for writing into the registers. Register 0 is hardwired to be always read as zero. Each register bit has a total of 12 transistors, four for the flip-flop and two extra for each port (see Figure 8). All ports are dual-rail; that is, they generate data and inverted data. Between reads and writes, the buses are precharged to a neutral state to reset the completion circuits and to prepare for the next transfer.

Read ports use a dual-ended sense amplifier (see Figure 9, next page). This sense amp detects a small difference between the true and false buses and drives them strongly in opposite directions, using transistors T5, T4, and T1. To work properly, the circuitry must select the register being read some time prior to applying the sense signal. To this effect,

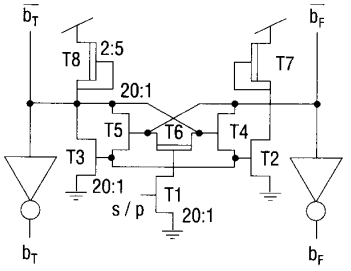


Figure 9. *Sense amplifier and precharge circuit for register file.*

we derive the sense signal $s/\overline{p}$ from an OR of all select signals for the given port.

When $s/\overline{p}$ is low, transistors T8, T3 (T7 and T2) precharge 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 circuit's symmetry. The buses are buffered before going into the data path, to isolate the sense amplifier and to restore the logic levels of the data signals.

The write ports generate a completion signal indicating that the write is done or that the buses have been precharged. We assume that the write is finished when all bits in the bus have valid data and one of the registers is selected. Likewise, the precharge is finished when all bits in the bus have been precharged and no register is selected. To generate the completion signal, we use a 17-input C-element, designed as in Figure 5.

Results

Using these techniques, we have designed several circuits and fabricated and tested them on the HGaAs II and HGaAs III processes offered by Vitesse Semiconductors. Among them are two small RAMs, a register file, and two microprocessors.

Asynchronous static memories. We designed and fabricated two types of static memories in GaAs. The first is a 16-word × 4-bit, dual-ported memory. Its purpose is to provide a small amount of fast memory so that the microprocessor can run test programs at full speed. Of 30 bonded devices, 29 were functional. The chip's access time is 5 ns, and it dissipates 500 mW at 2.2V.

The second SRAM has 64 words of 4 bits. We designed it in collaboration with H.P. Hofstee[10] as an intermediate step toward a larger memory to serve as a cache for the microprocessor. All 30 bonded devices received were functional. The access time is 3 ns, including pad delays, and the chip dissipates 700 mW at 2.3V.

We designed the 64×4 memory after we designed the first microprocessor, and we incorporated several improvements derived from our experience with the earlier design. We also carefully optimized the circuits for high speed and low power consumption. The performance we obtained indicates that the improvements we envision for the next microprocessor generation should be attainable.

Microprocessors. The first microprocessor design uses the circuits described by Tierno.[9] Our main concern was to get around parameter variation problems and noise margin considerations. We overcame these problems, but at the expense of power and performance. At 70 MIPS, the microprocessor consumes 4W.

The second microprocessor design incorporated the new ideas and circuits presented in this article. All of those circuits were fabricated and tested successfully.

We simulated the second design extensively with Hspice. The expected performance was about 200 MIPS with a dissipation of 2W. Power and speed predictions, using the Hspice models, have been very accurate so far. However, the measured performance is only 100 MIPS. The causes are still under investigation. We have found some evidence of fabrication problems and underestimation of the parasitic capacitances as extracted by the Magic layout program.

Another factor affecting performance is pad delay. So far, we have used ECL levels on the outside of the chip, to enable it to interface to standard parts and simplify prototyping. Pad delays are on the order of 1 ns, mostly spent in level conversion. The pad frame also uses a considerable amount of power—close to 2W under worst-case conditions for the microprocessor. Matched pad drivers and receivers in a system composed exclusively of GaAs parts would greatly reduce delay and power use. This would certainly be a requirement in the interface with cache memory.

In the course of this project, we designed several different GaAs circuits. To overcome the limitations of GaAs, most of them had very strict requirements. We found no general solution to the problem of synthesizing all logic circuits in a design as big as a microprocessor. Instead, we developed specific solutions for implementing completion trees, control circuits, PLAs, registers, and so on.

The first GaAs microprocessor, though disappointing in performance, gave us invaluable experience in verifying and testing GaAs circuits. Together with the RAM designs, it helped us design a second microprocessor with considerable performance improvements.

Porting the design of the original CMOS microprocessor was almost as easy as we expected. A few changes of the original CMOS design were necessary because of the complexity of register cells in the first GaAs version. We carried over these changes to the second microprocessor to speed up the redesign. Overall, the performance of the new microprocessor is satisfactory. At 50 MIPS/W, it offers remarkable speed for the power consumption.

In general, we are satisfied with our solutions to the problem of designing

asynchronous circuits in GaAs, although we were expecting better performance. The MESFET is far from an ideal switch; as a result, gates with good gain, noise, delay, and power characteristics are quite complicated. To make those gates more reliable, we must pay an overhead in delay and power, offsetting the raw performance gain of the GaAs MESFET over the silicon MOSFET.

More importantly, integration levels in GaAs are very limited compared to CMOS. An important source of performance in digital circuits is parallelism, which can more than make up for the difference in raw speed. Reduced integration levels greatly penalize circuits that can make efficient use of parallelism, such as microprocessors with wide data paths, or memories.

Of course, some problems are inherently sequential, such as serial communication channels. GaAs is an ideal technology for a router for a high-speed fiber optic network, or for a digital signal-processing chip for microwave circuits. For such problems, we can apply the tools of asynchronous design and carry over to GaAs all the advantages of asynchronous circuits. D&T

Acknowledgments

We are indebted to Marcel van der Goot, Steve Burns, Pieter Hazewindus, H. Peter Hofstee, Ray Milano, and Cindy Hibbert.

The research described in this article was sponsored by the Advanced Research Projects Agency, under ARPA order 6202, and monitored by the Office of Naval Research, under contract N00014-87-K-0745.

References

1. A.J. Martin, "Compiling Communicating Processes into Delay-Insensitive VLSI Circuits," *Distributed Computing*, Vol. 1, No. 4, 1986, pp. 226-234.
2. A.J. Martin et al., "The Design of an Asynchronous Microprocessor," in *Advanced Research in VLSI: Proc. Decennial Caltech Conf. VLSI*, C.L. Seitz, ed., MIT Press, Cambridge, Mass., 1989, pp. 351-373.
3. C.A.R. Hoare, "Communicating Sequential Processes," *Comm. ACM*, Vol. 21, No. 8, 1978, pp. 666-677.
4. A.J. Martin, "Synthesis of Asynchronous VLSI Circuits," in *Formal Methods for VLSI Design*, J.Straunstrup, ed., North-Holland, Amsterdam, 1990, pp. 237-283.
5. S.I. Long and S.E. Butner, *Gallium Arsenide Digital Integrated Circuit Design*, McGraw-Hill, New York, 1990.
6. K.R. Nary and S.I. Long, "GaAs 2-Phase Dynamic FET Logic—A Low-Power Logic Family for VLSI," *IEEE J. Solid-State Circuits*, Vol. 27, No. 10, 1992, pp. 1364-1371.
7. P.S. Lassen and S.I. Long, "Ultralow-Power GaAs-MESFET MSI Circuits Using 2-Phase Dynamic FET Logic," *IEEE J. Solid-State Circuits*, Vol. 28, No. 10, 1993, pp. 1038-1045.
8. A.J. Martin, "Asynchronous Data Paths and the Design of an Asynchronous Adder," *Formal Methods in System Design*, Vol. 1, No. 1, 1992, pp. 117-137.
9. J.A. Tierno, *Designing Asynchronous Circuits in Gallium Arsenide*, master's thesis, CS-TR-92-19, California Institute of Technology, Pasadena, Calif., 1992.
10. H.P. Hofstee, "Deriving Some Asynchronous Memories," 1991, unpublished, available from the author at California Institute of Technology, Dept. of Computer Science, Pasadena, CA 91125; or HPH@vlsi.cs.caltech.edu..

José A. Tierno received the engineering degree in electrical engineering from the Universidad de la República, Montevideo, Uruguay, and an MSc degree in electrical engineering from the California Institute of Technology, where he is currently working toward a PhD in computer science. His research concentrates on low-energy asynchronous design. Other interests include computer architecture and GaAs VLSI design.

Alain J. Martin is a professor of computer science at the California Institute of Technology. He graduated from the Institut National Polytechnique de Grenoble. His research interests include concurrent and distributed programming and its application to the design of VLSI circuits and highly concurrent computing systems.

Drazen Borkovic received the MSc degree in electrical engineering from the California Institute of Technology.

Tak Kwan (Tony) Lee received the BS degree in computer engineering and the BA degree in mathematics from UC San Diego. He received the MS degree in computer science from the California Institute of Technology, where he is working on his doctoral thesis on analyzing the performance of delay-insensitive circuits. His other research interests include CAD tools and asynchronous arithmetic units.

Send correspondence to José A. Tierno, Caltech, Dept. of Computer Science, Pasadena, CA 91125; or jat@vlsi.cs.caltech.edu.