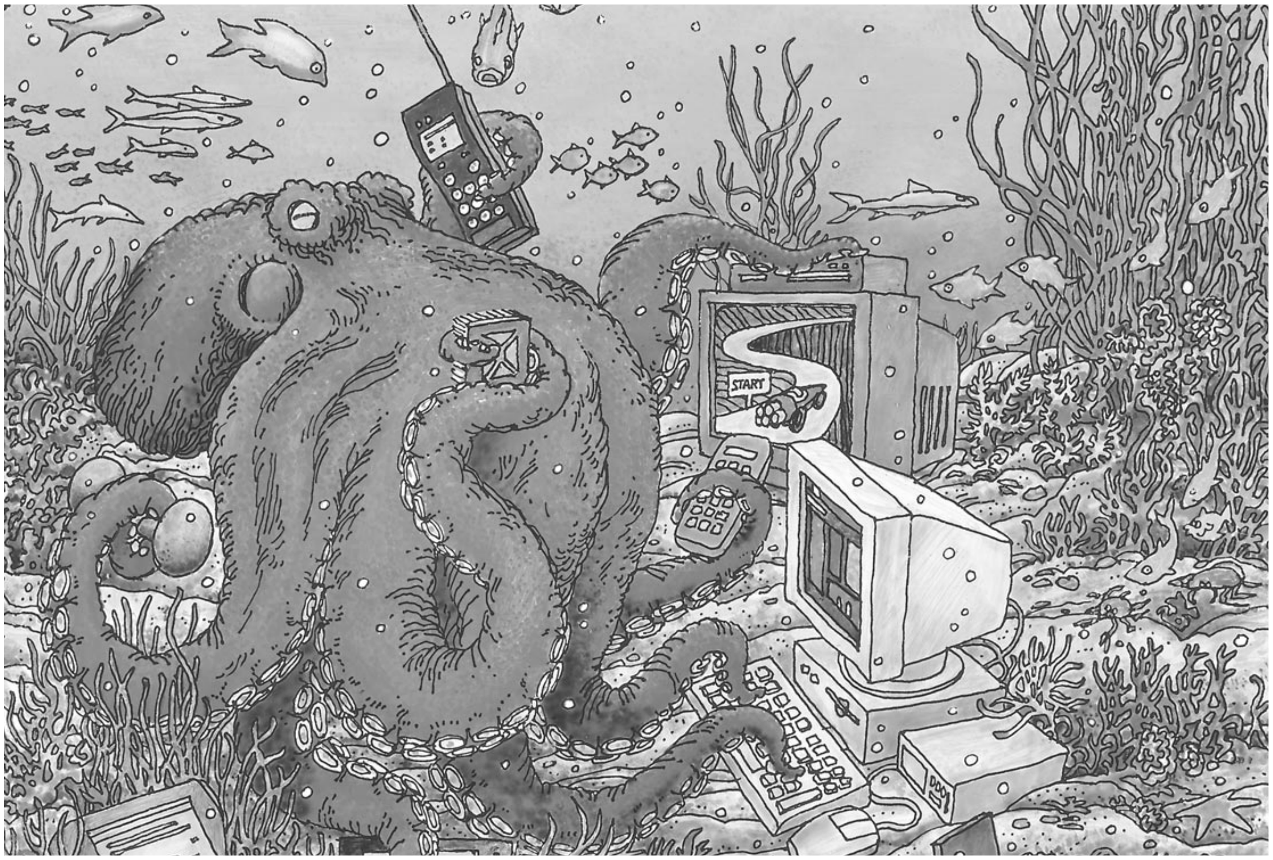

Guest Editor's Introduction

ARM ARCHITECTURE AND SYSTEMS

Dave Jaggar

Advanced RISC Machines

Although the articles in this issue of *IEEE Micro* share a common theme, you will find the collection diverse. The technology surrounding ARM's processor line led to articles spanning topics from detailed microprocessor design to end products powered by the processor.

ARM designs microprocessors and related technology and licenses them as intellectual property. It does not manufacture silicon, relying instead on semiconductor partners to manufacture, market, and sell solutions based on ARM products. The clear role distinction between ARM and its business partners allows true partnership; we do not compete with our licensees.

ARM licenses its processors to industry-leading manufacturers including Alcatel, Atmel, Asahi Kasei Microsystems, Cirrus Logic, Digital, GEC Plessey, Hyundai, Lucent, Lucky Goldstar, NEC, OKI, Philips, Rockwell, Rohm, Samsung, Sharp, Sony, Symbios, Texas Instruments, VLSI, Yamaha, and others still to be announced. Some of these companies use the processors for very specialized applications. However, most use them for products such as mobile telephones, automotive engine management systems, PostScript laser printers, or global positioning systems. All these products require high-performance, low-cost, low-power-consumption computing.

0272-1732/97/$10.00 © 1997 IEEE

The ARM processor cores

The ARM7 is a small, 32-bit microprocessor with very low power consumption. A three-stage pipeline occupies minimal silicon area yet allows division of the execution time of each instruction into three parts: instruction fetch from memory, instruction decode, and instruction execution. The instruction execution stage is the most complex. Register read, a shift applied to one operand, an ALU operation, and finally a register write all execute in one clock cycle. This limits the processor's maximum clock speed to around 80 MHz on a 0.35-micron silicon process. However, that speed is more than enough for the cost-sensitive applications using ARM7.

The combined shift and ALU execution stage is also an important ARM feature. A single instruction can specify one of its two source operands for shifting or rotation before it is passed to the ALU. This allows very efficient bit manipulation and scaling code, and virtually eliminates single shift instructions from ARM code. (The ARM processor does not have explicit shift instructions; a move instruction applies a shift to its operand.)

ARM7 also uses a von Neumann memory architecture; the instructions and data occupy a single address space and are accessed with individual address and data buses. Though this limits performance—instruction fetching (and hence execution) must stop for instructions that access memory—the reduced cost of a single memory outweighs performance in many embedded applications. To reduce the penalty of data accesses stalling the pipeline, ARM implements load multiple and store multiple instructions. These instructions can move any of the ARM registers to and from memory, and update the memory address register automatically after the transfer. This not only allows one instruction to transfer many words of data (in a single bus burst), it also reduces the amount of instructions needed to transfer data. As a result, ARM code is smaller than other 32-bit instruction sets.

Although the pipeline stalls during load and store operations, the ARM7 can continue useful work. These instructions can specify an update of the base address register with a new address after (or even before) the transfer. RISC architectures would normally use a second instruction (add or subtract) to form the next address in a sequence. ARM does it automatically with a single bit in the instruction, again a useful saving in code size.

The ARM instruction set has one further useful feature. Most architectures have conditional branch instructions. These follow a test or compare instruction to control the flow of execution through the program. Some architectures also have a conditional move instruction, allowing data to be conditionally transferred between registers. The ARM instruction set takes this functionality to its logical extreme, allowing all instructions to be conditionally executed. Loads, stores, procedure calls and returns, and all other operations may execute conditionally after some prior instruction to set the condition code flags. (Any ALU instruction may set the flags.) This eliminates short for-

Establishing a new microprocessor architecture in the world marketplace is not an everyday occurrence. ARM's success comes largely from its ability to break industry rules and find new, more efficient solutions to existing problems. For example, when faced with the trade-off between meeting customer demands for less program memory usage and still delivering maximum performance for critical applications, ARM designers produced the Thumb processor. Thumb processors have two instruction sets, one to deliver maximum performance, the other to provide minimal code size.

When faced with the need to execute DSP algorithms, designers added more than a multiply-accumulate instruction. They added the Piccolo coprocessor, which transforms ARM's processor into a highly efficient DSP engine that performs microprocessing and very high performance signal processing. A single operating system controls both, and Piccolo-based solutions were engineered under a single code development and debugging environment.

A wide range of internal and third-party tools support ARM's processors, forming a large software and development infrastructure. These include development tools such as debuggers, C++ compilers, in-circuit emulators and development cards, real-time operating systems, and low-level driver software to high-level application software. Accelerated Technology, Enea OSE Systems, ISI, JavaSoft, JMI, Microtec, Microsoft, Perihelion, Psion, Wind River, and others provide operating systems and their corresponding development tool chains.

ARM currently provides three basic processor cores, with two more in development, as described in the above box.

The six articles in this issue span the entire ARM line, from insights into the heart of ARM's low-power techniques to presentations of complex single-chip, portable organizer products.

The first article, on the ARM7TDMI, highlights many of the ARM techniques for delivering very low power consumption. The analysis of the processor core's power consumption is in respect to both silicon process and functional blocks within the processor.

The AMBA article describes the ARM bus that connects both high-speed devices such as caches and processors, and lower speed, low-power devices such as peripherals. The author offers a solution to the conflicting demands of high performance and low power consumption and describes a uniform test strategy with small die size.

Cross-development for embedded systems is the subject of the next article. It discusses the problems of modeling a processor core in both HDL and software development environments, and presents ARM's current and future solutions to this problem.

An article on software modems describes the demands of a V.34 modem upon various ARM processors. Designers normally use custom hardware to implement a high-speed modem. However, the power of modern microprocessors ward branches in ARM code. Once again, this improves code density and avoids flushing the pipeline for branches, increasing execution performance.

ARM8 is the next core in the ARM line. It extends the ARM7 implementation in two fundamental ways: two additional pipeline stages and a new cache interface. ARM7's execute stage splits into three separate stages on ARM8, and register read moves back into the decode stage. The two additional pipeline stages perform memory accesses and register writes. Because each instruction executes over multiple cycles, register-forwarding paths must pass data between successive instructions. This is necessary because one instruction will not have written its result to the register file before the next two instructions have read their source register values.

ARM8 incorporates a single cache interface that allows instruction fetches in parallel with data accesses. It retains ARM7's von Neumann cache interface, but doubles the bandwidth of the interface to provide 64 bits every cycle. ARM8 also uses a sophisticated prefetch buffer and branch prediction unit to fetch instructions ahead of the execution unit. On every cycle, one instruction is fed to the processor from the prefetch buffer. When the cache is not in use for a data access, two instructions are loaded into the prefetch buffer. This allows the single cache to satisfy both data and instruction accesses.

ARM8 behaves similarly in performance to a Harvard machine with separate instruction and data caches, yet retains the simplicity of a single cache machine. Static branch prediction predicts the target of branch instructions; backward branches are assumed taken (loops) and forward branches untaken (conditional code). Correctly predicted branches do not enter the main execution engine and thus effectively execute in zero cycles. Mispredicted branches take three cycles to correct. ARM8 delivers 100-MHz operation in a typical 0.35-micron process, and lowers the average number of clock ticks per instruction to around 1.5. This increases overall performance by about 70% over ARM7.

Digital Equipment Corporation codesigned the StrongARM1, the fastest of our current processors. Adoption of a Harvard architecture to deliver maximum cache throughput and a five-stage instruction pipeline to allow maximum clock rate produced an embedded processor that is faster than some workstation processors. StrongARM110 incorporates two 16-Kbyte caches maintained even when the processor is coupled to a relatively low-speed memory system. When coupled with Digital's very fast 0.35-micron process, which operates with a 2-volt supply, StrongARM1 machines deliver 233 MHz. With less than 1 Watt of power consumption, this makes the StrongARM power consumption/performance ratio the best in the industry.

Designers are also working on ARM9 and StrongARM2 designs. These products will extend the performance to new levels and application domains at embedded-class cost. ARM plans to launch these products late in 1997.

allows implementation purely in software, using only a small proportion of the total processor horsepower.

Two authors from Palmchip Corporation describe the process of using the ARM processor in a complex ASIC. They explain the designers' choice of the processor as well as detailing other design issues.

Finally, the ARM7100 article discusses using a sophisticated single chip, designed by ARM, as a solution to many handheld, low-power-consumption devices. The authors disclose two new handheld products, describing the different applications for the processor.

WHAT DOES THE FUTURE of the ARM microprocessor look like? Expect to see more industry-leading performance, low power consumption, and small die size, as well as new extremes in code density, specific application performance, and ultra large scale integration. ARM's ability to deliver maximum efficiency will encompass an even broader range of applications at the crossroads of portable, consumer, and embedded control applications.

Dave Jaggar is director of ARM's Austin Design Center, where he leads future ARM processor design. He is the architect of the Thumb instruction set and co-architect of Piccolo. His interests lie in next-generation computer architectures, and their implementations, compilers, and new languages.

Jaggar holds an MSc degree in computer science from Canterbury University, New Zealand.

Direct comments about this issue to Dave Jaggar, Advanced RISC Machines Ltd., Austin Design Center, Building 3, Suite 560, 1250 Capital of Texas Highway, Austin, TX 78746; djaggar@arm.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 150 Medium 151 High 152