Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

ARC 700 Secrets Revealed

It's the First ARC Processor for High-End Embedded Operating Systems

By Tom R. Halfhill {6/21/04-01}

Challenging ARM in the embedded-processor market is as daunting as challenging Intel in the PC market: you're cruisin' for a bruisin'. No wonder ARC International wanted to delay revealing everything about its new ARC 700 processor core until a marketing plan was in place. As ARC recently disclosed at **Embedded Processor Forum 2004**, the ARC 700 has additional features that directly challenge ARM's most popular processor cores.

Microprocessor Report covered the ARC 700 in depth earlier this year, detailing its new seven-stage pipeline, higher clock speeds, DSP instructions, dynamic branch prediction, wider cache interfaces, single-cycle adder, nonblocking load/store pipeline, out-of-order completion, and new support for multicore SoCs. (See *MPR 3/8/04-01*, "ARC 700 Aims Higher.") Those features alone justify the ARC 700's ground-up redesign. At EPF, ARC further revealed that the ARC 700 is the company's first processor capable of running Linux and other sophisticated embedded operating systems. The reason: it's the first ARC processor with a memory-management unit (MMU), translation lookaside buffer (TLB), precise exception model, and multiple privilege levels.

Virtual memory, precise exceptions, and the ability to segregate kernel-level code from user-level tasks are critical features in higher-end embedded processors. Without those features, a processor can run a variety of real-time operating systems, but it cannot run the most powerful operating systems for advanced embedded applications. Adding those capabilities to the ARC 700 makes it more suitable for the role of a host processor in networking equipment, communications infrastructures, and consumer electronics. It also puts the ARC 700 in direct competition with similar processor cores from ARM and MIPS Technologies. For the first time, ARC is breaking out of the deeply embedded market and going head-to-head against the best 32-bit processors from its most formidable competitors.

ARC says the first release of an embedded Linux for the ARC 700 will be available in July, though the company hasn't yet announced the vendor. The first release is being ported with GNU software-development tools. GNU C isn't as optimized for the ARC 700 as ARC's own MetaWare tools, so a second release—recompiled with the MetaWare tools—is scheduled for later this year.

Meanwhile, ARC is negotiating with other vendors to port additional operating systems to the ARC 700. ARC is mum on the subject, but SymbianOS and Wind River's VxWorks AE are possibilities. Microsoft's Windows CE.NET would be a huge win, but ARC probably won't undertake such a large investment at this stage. Like other ARC processors, the ARC 700 can run a variety of real-time embedded operating systems—such as Precise/MQX, ThreadX, and the regular version of VxWorks—that don't need virtual memory, precise exceptions, and special privilege modes.

ARC 700 Has Supercharged Microarchitecture

Adding support for heavy-duty operating systems required ARC to extensively overhaul a microarchitecture that had remained largely unchanged for three processor generations, going back to the ARC 600, ARCtangent-A5, and ARCtangent-A4. However, the other improvements to the ARC 700 (mentioned above) were equally dramatic, so a ground-up redesign was still necessary. The ARC 600, introduced in 2003, remains in the product line as a smaller, simpler, lower-power alternative to the ARC 700. (See *MPR 12/15/03-01*, “ARC Alters Trajectory.”)

Supporting virtual memory requires, at minimum, an MMU to map physical memory addresses to logical addresses. ARC’s MMU is a straightforward design that manages a 32-bit virtual address space and 32-bit physical addressing, for up to 4GB of system memory. Program code and data share the same unified address space, although the MMU can manage separate regions of protected memory for code and data within the address space. The MMU is a user-selectable option in the ARChitect 2 processor-configuration tool, so SoC designers who don’t need its features can omit it, saving about 20,000 gates.

Previous ARC processors had a user-configuration option for either a Harvard bus architecture (separate 32-bit I/O buses for fetching instructions and data) or a von Neumann architecture (a single 32-bit I/O bus for fetching instructions and data). The ARC 700 lets users define up to five I/O interfaces, 32 or 64 bits wide, for an instruction cache, data cache, closely coupled instruction memory, closely coupled data memory, and DSP X and Y data memories. Interface protocols are also user configurable: AMBA, BVCI, or ARC’s own proprietary protocol. It’s still possible to configure a minimal version of the ARC 700 that has shared memory for instructions and data without caches, but the ARC 600 is usually a better choice for designs that don’t need the higher-end features of the ARC 700.

Memory pages on the ARC 700 are fixed in size at 8KB. It’s not uncommon for an embedded processor to have fixed-size pages, partly to save gates and partly because memory subsystems in embedded applications tend to be small. The ARC 700’s fixed page size of 8KB is twice as large as the 4KB pages commonly found in some other embedded processors.

However, the directly competing processors from ARM, MIPS, and Tensilica are more flexible in this regard. ARM1136 and ARM1176 processors allow programmers to define memory pages ranging in size from 4KB to 16MB, and even the smaller ARM926EJ-S can define pages from 1KB to 1MB. (The latest ARM1156-series cores lack an MMU, so they don’t support memory paging at all.) Some MIPS32 processors offer even more flexibility: the 4KEc core supports page sizes from 1KB to 256MB, and the 24K core supports pages from 4KB to 256MB. Likewise, Tensilica’s Xtensa V supports page sizes from 4KB to 256MB. (The new Xtensa LX doesn’t work with Tensilica’s MMU yet, so it doesn’t support memory paging.)

ARC notes that large memory pages aren’t required for Linux, the operating system for which the MMU is primarily designed. A basic port of embedded Linux usually defines one page size. The ARC 700’s fixed 8KB memory pages are sufficient for now, but this is one opportunity for improvement in future releases.

The Biggest TLB in Its Class

The ARC 700 has an impressively large primary TLB, with 256 entries. It is supplemented by a pair of micro-TLBs: a four-entry buffer for instructions and an eight-entry buffer for data. However, loading entries into the primary TLB involves some circumlocution. The TLB control registers are in the processor’s auxiliary register space, which requires special load/store instructions to access. Therefore, the processor must execute two 32-bit loads to fetch a 64-bit page-table descriptor from memory into a pair of general-purpose registers, then execute two store-to-auxiliary instructions to move the descriptor into the TLB. ARC says the MMU has mechanisms to minimize the effect of this indirection when the TLB’s miss handler loads a new entry.

Among the page-descriptor fields is a global bit that allows a memory page to appear in every virtual-address space; a cache bit that determines whether a page can access the instruction and data caches; and six protection bits that allow a page to have separate read, write, and execute permissions for user- and kernel-level processes. An 8-bit address-space identifier (ASID) allows the operating system to manage up to 256 processes in the TLB. Additional simultaneous processes are possible, but normally only the most active or recently used processes will be buffered in this manner.

To save gates, the large primary TLB is two-way set-associative, not fully associative. However, the micro-TLBs are fully associative. ARC says this TLB hierarchy was the best trade-off between performance and gate count, because most programs will find the page descriptors they need in the micro-TLBs first, whereas a fully associative primary TLB would have required more gates than the micro-TLBs do. Of course, this is a judgment call that depends on the memory-access patterns of the software, but it’s the same reasoning that justifies greater set-associativity in L1 caches than in L2 caches. In any case, the MMU’s programming interface is independent of the TLB implementation, so future processors can introduce larger and faster TLBs without breaking any software.

Compared with the TLBs in similar embedded processors, the ARC 700’s TLBs stack up well. Its 256-entry primary TLB is much larger than the primary TLBs in the ARM926, ARM1136, ARM1176, and MIPS32 cores, which have as few as eight entries and no more than 64. Tensilica’s Xtensa V and Xtensa LX processors have no primary TLBs at all, and the ARM926EJ-S has no micro-TLBs. The micro-TLBs in the ARM11, MIPS32, and Xtensa processors are slightly larger than those in the ARC 700, but the differences are minor. Table 1 compares these and other features of the ARC 700 with the features of similar 32-bit embedded-processor cores from other vendors.

ARC’s trade-off is that its huge primary TLB isn’t fully associative, unlike the primary TLBs in the ARM1136, ARM1176, MIPS 4KEc, and MIPS 24K cores. The ARC 700’s two-way set-associative TLB is a simpler design that saves gates and can refill a micro-TLB in only five clock cycles, but a TLB miss requires a time-consuming page-table walk to refill the buffer. Of course, with 256 entries, that will happen less often. All things considered, ARC's TLB hierarchy makes a good trade-off between buffer capacity and gate count.

User Programs Can't Touch Kernel Processes

In previous ARC processors, any program had full access to the machine, in effect running free in "kernel mode"—the only mode. That's good enough for embedded applications that run only one or a few trusted tasks, but in more sophisticated systems, it's like turning a three-year-old child loose in a Fabergé egg shop. In the ARC 700, the previous anything-goes execution mode is now the higher-privilege kernel mode, and a new lower-privilege mode is for user tasks. This addition required several changes to the ARCompact instruction-set architecture (ISA).

Two core registers that store the return address from an interrupt—ILINK1 and ILINK2—are now off limits to user-level programs. The other core registers remain available, including general-purpose registers R0–R28, the link register (R31), the loop-counter register (R60), and the program counter (R63). All but three of the primary auxiliary registers are also off limits to user programs. (The ARC ISA supplements the configurable core-register file with a configurable auxiliary-register file, which has a huge 32-bit address

	ARC	ARC	ARM	ARM	ARM	ARM	MIPS	MIPS	Tensilica	Tensilica
Feature	**ARC 700**	**ARC 600**	**926EJ-S**	**1156T2F-S**	**1176JZF-S**	**1136JF-S**	**4KEc Pro**	**24K Pro**	**Xtensa V**	**Xtensa LX**
Architecture	ARCompact	ARCompact	ARMv5TEJ	ARMv6T2	ARMv6Z	ARMv6	MIPS32 R2	MIPS32 R2	Xtensa	Xtensa
Configurability	High	High	Low	Low	Low	Low	Medium	Medium	High	High
Core Frequency [1]	400MHz	290MHz	220–250MHz	333–550MHz	333–550MHz	333–550MHz	255–300MHz	400–550MHz	350MHz	350MHz
Pipeline Depth	7 stages	5 stages	5 stages	9 stages	8 stages	8 stages	5 stages	8 stages	5 stages	5–7 stages [5]
DMA Controller	—	—		Yes	Yes	Yes	—	—	—	—
MMU	Yes	—	Yes	—	Yes	Yes	Yes	Yes	Optional	—
Page Sizes	8KB	—	1KB–1MB	—	4KB–16MB	4KB–16MB	1KB–256MB	4KB–256MB	4KB–256MB	—
Primary TLB	256 entries, 2-way assoc	—	64 entries, 2-way assoc; 8 entries, fully assoc	—	8 entries, fully assoc; 64 entries, 2-way	8 entries, fully assoc; 64 entries, 2-way	16 entries, fully assoc	16–64 entries [5], fully assoc	—	—
Micro TLB (Instructions)	4 entries, fully assoc	—	—	—	10 entries, fully assoc	10 entries, fully assoc	4 entries, fully assoc	4 entries, fully assoc	8–32 entries [5], 1–8-way assoc [5]	—
Micro TLB (Data)	8 entries, fully assoc	—	—	—	10 entries, fully assoc	10 entries, fully assoc	4 entries, fully assoc	8 entries, fully assoc	8–32 entries [5], 1–8-way assoc [5]	—
DSP Extensions	Yes	Optional	Yes	Yes	Yes	Yes	—	—	Optional	Optional
FPU	—	—	—	Yes [6]	Yes [6]	Yes [6]	—	Optional	Optional	Optional
Java Extensions	—	—	Yes	—	Yes	Yes	—	—	—	—
16-Bit Instr [2]	Yes	Yes	Thumb-1	Thumb-2	Thumb-1	Thumb-1	MIPS16e	MIPS16e	Yes	Yes
Branch Predict	Dynamic	Static	—	Static/ Dynamic	Static/ Dynamic	Static/ Dynamic	—	Dynamic	—	—
Privilege Levels	2	1	2	2	3	2	4	4	4	4
Task Security	—	—	—	—	TrustZone	—	—	—	—	—
Gate Count[3]	95KG	27KG	n/a	n/a	n/a	n/a	32KG	n/a	18KG	20KG
Power / MHz [4]	0.15mW	0.04mW	0.35mW	0.6mW	0.6mW	0.6mW	0.12–0.37mW	0.5mW	0.04mW	0.04mW
Embedded Linux	July 2004	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes	—
Palm OS	—	—	Yes	—	—	—	—	—	—	—
SymbianOS	—	—	Yes	Yes	Yes	Yes	—	—	—	—
Windows CE	—	—	Yes	Yes	Yes	Yes	Yes	Yes	—	—
Core Availability	Feb 2004	2003	2001	3Q04	Jun 2004	2002	2003	Apr 2004	2002	Jun 2004

Table 1. All the processors in this table are 32-bit synthesizable RISC cores available as licensable intellectual property (IP), and all have been described in previous issues of *MPR*. [1]Core clock frequencies are targets for a 0.13-micron CMOS process and will vary according to the final core configuration, synthesis script, synthesis library, fabrication process, and other factors. [2]All these processors supplement their 24- or 32-bit standard instruction sets with 16-bit instructions for greater code density. [3]Gate counts are based on vendor estimates for a base configuration with no caches. [4]Power-consumption figures are based on vendor estimates for a base configuration. [5]Design-time configurable option. [6]These ARM11 cores are also available without FPUs. n/a: not available.

space. The ISA defines about three dozen primary auxiliary registers for internal use, leaving the rest of the address space available for user-defined registers.) The primary auxiliary registers accessible to user tasks are LP_START and LP_END (for loop addresses) and an auxiliary view of the program counter (mainly for debugging). In addition, user tasks can access the processor's status flags and the mode bits for multiply-accumulate (XMAC) instructions.

As with previous ARC processors, the ARC 700 supports two priority levels of interrupts. However, all interrupts now force the processor into kernel mode, and user-mode programs cannot access the interrupt state. All interrupts are disabled while the kernel is servicing an exception, but the processor can take an exception while servicing an interrupt. Several new status registers and flags store information about interrupts, such as the cause and the priority level. There's also a new instruction—RTIE, return from interrupt/exception—that exits from either type of interrupt, depending on the machine state.

Precise exceptions allow the ARC 700 to commit the results of all instructions preceding the instruction that triggered the fault; abort the faulted instruction before committing its result; notify the exception handler as to what caused the exception; and restart the instruction pipeline where it left off after returning from the handler. Exceptions trigger immediately before the faulted instruction reaches the final writeback stage of the seven-stage pipeline. If the faulted instruction would not have completed anyway—for example, if it follows a mispredicted branch or another exception—the processor ignores the exception and aborts the instruction. The ARC 700 has several new status registers and flags to store various exception-related state, such as the fault address, the return address, and the cause of the exception.

The ARC 700's new exception model and privilege modes are not radically different from the implementations in other embedded processors, but they are an important step for ARC. When combined with the other new kernel-level features—virtual memory, paging, memory protection—they make the ARC 700 seem like a grown-up processor, capable of tackling higher-end embedded applications.

ARM, MIPS, and Tensilica still have some advantages. New ARM1176 processors based on the ARMv6Z ISA have a special execution mode called TrustZone that allows security-related tasks to protect their code and data against tampering. (See *MPR 8/25/03-01*, "ARM Dons Armor," and *MPR 1/5/04-01*, "ARM Expands ARM11 Family.") Some ARM, MIPS, and Tensilica processors have three or four privilege levels, which allows different system- or user-mode tasks to enjoy different privileges.

ARC's goal was to significantly expand the capabilities of its processor architecture while maintaining a reasonable gate count. Overall, the ARC 700 strikes a good balance. However, *MPR* believes security is becoming such a large concern that CPU-level features like TrustZone will someday be as commonplace as MMUs and TLBs, even in low-power embedded processors.

New Instructions Improve ARCompact ISA

As noted in our previous report on the ARC 700, 13 DSP instructions that are optional on previous ARC processors are now standard equipment on this processor. (The XMAC instruction and the DSP XY data memories remain optional.) The ARC 700 has nine additional new instructions, only two of which were publicly disclosed before EPF. Some of the new instructions support the features described above; others provide additional capabilities. Table 2 lists all nine new instructions.

RTIE, mentioned previously, is a single instruction that returns from either an interrupt or an exception. EX and SYNC, previously disclosed, are useful for multitasking and multiprocessing. EX is an atomic (uninterruptible) operation that exchanges a value in memory with the contents of a register. SYNC is for synchronizing the instruction and data caches of two or more ARC 700 cores on a multiprocessor chip—it pauses execution until all pending cache fills and flushes have completed and all outstanding load, store, and atomic-exchange instructions are completed.

TRAP0 (32-bit) and TRAP_S (16-bit) are new instructions required by the new privilege levels. They call the operating system from user mode, usually to switch control to a kernel-mode exception handler. (TRAP0 replaces the SWI instruction in previous ARC processors.) SLEEP stops the clocks to save power while the processor is idle; previous ARC processors have a SLEEP instruction, but this version executes in kernel mode only. The BRK (32-bit) and BRK_S (16-bit) instructions halt execution, usually to insert a breakpoint for debugging software. They aren't entirely new, either, but in the ARC 700, they're

Instruction	Description	Notes
BRK	Software breakpoint (32-bit)	Available only in kernel mode except with external debugger
BRK_S	Software breakpoint (16-bit)	Same as BRK for 16-bit ARCompact instructions
EX	Atomic exchange	Exchange value in memory with register (uninterruptible)
PF	Prefetch data	Read line from data cache at specified address
RTIE	Return from interrupt or exception	Single instruction returns from either level of interrupt
SLEEP	Enter low-power sleep mode	New version for kernel mode only
SYNC	Synchronization	Instruction/data synchronization in multiprocessors
TRAP0	Call operating system (32-bit)	Switch control to kernel-mode exception handler
TRAP_S	Call operating system (16-bit)	Switch control to kernel-mode exception handler

Table 2. These nine instructions are new on the ARC 700. Before EPF, only the EX and SYNC instructions were publicly disclosed. The ARC 700 also has 13 standard DSP instructions that are optional on other ARC processors; see Table 1 in our previous ARC 700 article, *MPR 3/8/04-01*, "ARC Aims Higher."

valid only in kernel mode, unless they are enabled in user mode by an external debugger.

That leaves PF (prefetch), which reads a line of data into the data cache from the specified memory address. (The line size, like other cache parameters, is customizable at design time with the ARChitect 2 processor-configuration tool.) PF allows programmers to explicitly load data into the cache instead of relying on the processor's automatic caching policy. However, unless the program locks the cache line after filling it, there's no guarantee the prefetched data will be available the next time the program reads the cache—the processor may evict the data, according to the cache's normal replacement policy.

An existing instruction called LP has been improved in the ARC 700. LP (and its conditional counterpart, LPcc) sets the starting and ending addresses of a loop. Previously, setting up a single-instruction loop required using the SR (set loop register) instruction in addition to LP. On the ARC 700, LP can set up any kind of loop without using SR. Although this seems like a miniscule improvement, ARC says it's now easier for compilers to create tight single-instruction loops—especially zero-overhead loops in DSP routines. Furthermore, the enhanced LP instruction closes a potential security loophole (pun intended) that might cause a loop inside an interrupt or exception handler to jump out of the handler to a bogus address in the loop registers. As with previous ARC processors, the ARC 700 can enter a loop explicitly, or—with a conditional instruction—implicitly.

There's nothing earthshaking about the nine new instructions in the ARC 700, but they are useful improvements to an ISA that has been steadily evolving for ten years. ARC's philosophy is to keep the base ISA as small as possible to conserve gates, silicon, and power. Unlike most other architectures, the ARCompact ISA is highly configurable and extendable. Customers can add new 32-bit instructions, 16-bit instructions, conditional instructions, core registers, auxiliary registers, status flags, and almost anything else they can think of. The base ISA is purposely designed for minimal functionality in general-purpose embedded applications, leaving application-specific customization as an exercise for the SoC developer.

Growing Up, With Room to Grow

One consequence of growing up is growing larger. The base configurations of previous ARC processors were admirably small for 32-bit cores, ranging in size from about 16,000 gates to 27,000. The ARC 700—even without caches, DSP XY memories, and optional MMU—weighs in at about 95,000 gates. It rapidly grows heavier with the addition of calorie-rich features: 140,000 gates with 16KB caches and a 32×32-bit multiplier; 160,000 gates after adding the MMU; and 220,000 gates after adding the DSP XY memories and XMAC instruction.

Those gate counts assume the synthesis compiler is optimizing for clock speed, targeting core frequencies of 300–450MHz in a 0.13-micron process. When optimized for silicon area, the base configuration shrinks to 66,000 gates, as Figure 1 shows. With 16KB caches and the 32-bit multiplier, the area-optimized ARC 700 contains about 105,000 gates. Even so, the gate counts explain why the 27,000-gate ARC 600 remains in the product line as a lower-power alternative.

Although the ARC 700 seems rotund alongside the ARC 600, it's a sleek athlete when compared with similar processor cores from other companies. ARM doesn't publicly release gate counts for its cores, but an ARM1136 or ARM1176 processor without caches will require about 4.45mm^2 of silicon and consume 0.6mW per megahertz. That's 3.6 times the size and 2.4 times the power consumption of a fully configured ARC 700 with 16KB caches, 32-bit multiplier, MMU, and DSP extensions (1.23mm^2, 0.25mW per megahertz). Even the ARM926EJ-S configured without caches requires about 2.2mm^2 of silicon and consumes 0.35mW per megahertz, which makes the ARC 700 seem tiny.

Likewise, the MIPS32 24K core is 2.4 times as large as the ARC 700 and consumes twice as much power. The MIPS32 4KEc core is a better match for the ARC 700 in silicon area and power, but its shorter pipeline restricts it to lower clock speeds.

Some missing features explain why the ARC 700 is much smaller than ARM11-family processors. Most notably,

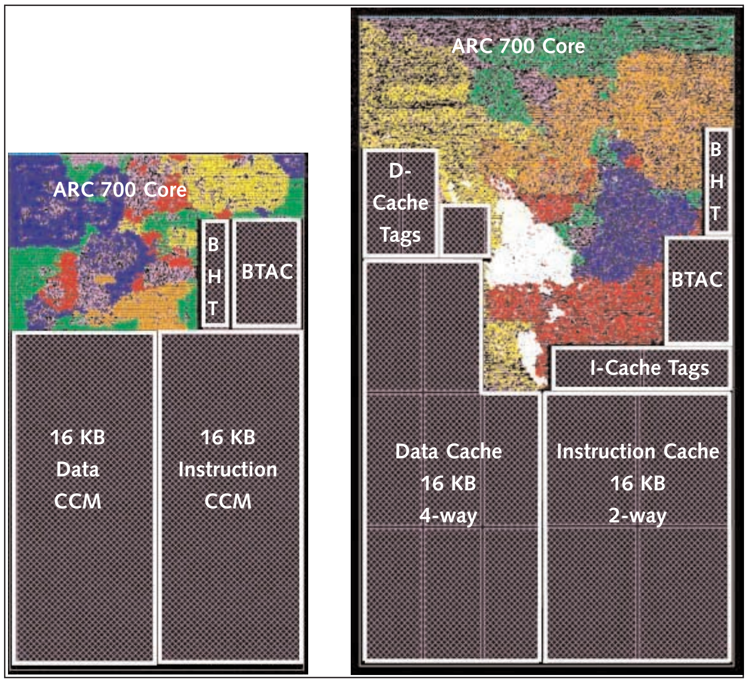

Figure 1. Synthesis scripts really do make a difference. When synthesized for higher clock speed (right), an ARC 700 core with caches can hit 420MHz, but it requires 140,000 gates and 0.78mm^2 of silicon in a typical 0.13-micron process. When synthesized to conserve silicon, and with closely coupled memories (CCM) instead of caches (left), the ARC 700 requires only 66,000 gates and 0.37mm^2 of silicon, but it runs at 120MHz.

Price & Availability

The ARC 700 with DSP extensions is available now for IP licensing as a fully synthesizable Verilog model. Up-front license fees and royalties are negotiable and not publicly disclosed. All development tools and peripheral IP available for the ARC 600 are compatible with the ARC 700 and are available now. For more information: *www.arc.com*.

the ARC 700 lacks an FPU, a DMA controller, and extensions for accelerating Java programs. The ARM11 also has a slightly deeper pipeline, so it can reach somewhat higher clock frequencies. Another difference is TrustZone, though it probably contributes little to the ARM11's gate count.

The MIPS 24K can reach higher clock speeds than the ARC 700, but at the expense of much higher power consumption. An FPU is optional for the 24K and would increase power consumption even further.

Overall, the ARC 700 is easily the best ARC processor to date. It adds many new features that reflect the maturation of the architecture and ARC's intention to challenge the best embedded-processor cores from industry leaders. Indeed, the ARC 700 takes such a big step beyond the ARC 600 and earlier ARC cores that its few shortcomings are entirely forgivable. It's already a fair match against the competition and is sure to grow even more competitive in the future. ◇