Tools and Techniques Surmount the Multi-core Challenge

Jon Titus, Senior Technical Editor

Tools and Techniques Surmount the Multi-core Challenge

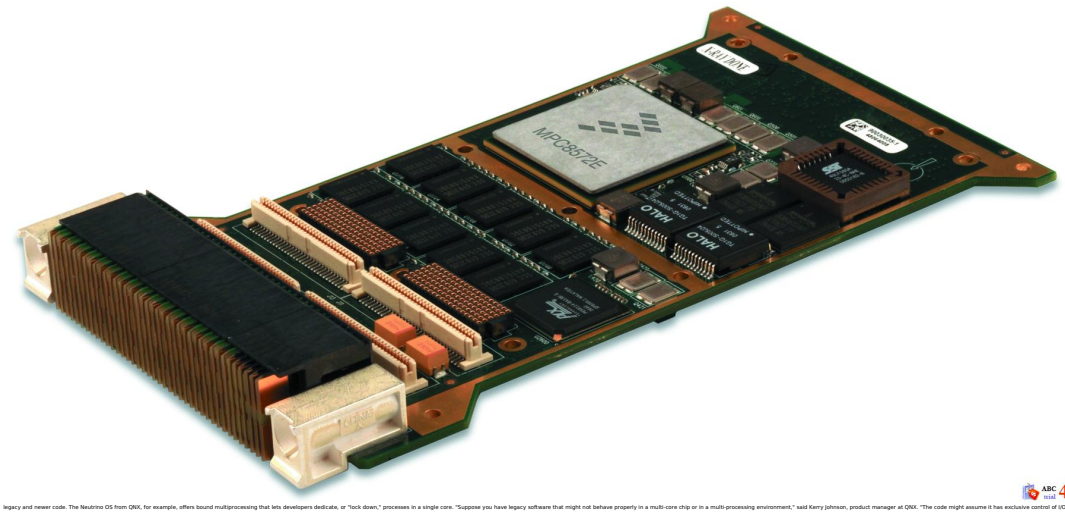
It's easier than you think to put a multi-core IC into a design.

by Jon Titus, Senior Technical Editor

Multi-processor computers have existed for some time, but only within the last few years have engineers had the opportunity to buy off-the-shelf chips with more than one processor or "core." These devices come in two varieties; symmetrical and asymmetrical. The first group provides multiple "clones" of the same core CPU, thus the term symmetrical. The second group includes devices that put different types of CPUs, DSPs, and accelerators in a system on a chip. I'll concentrate on the former multi-core technologies.

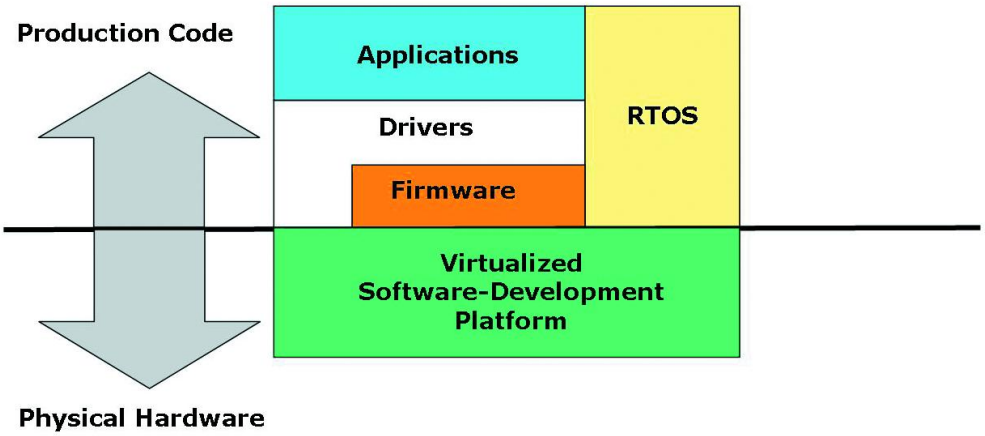
According to Rob Scidmore, chief executive officer at Extreme Engineering Solutions, few differences exist in the hardware used with a single processor and with a multi-core processor. "The challenges arise when you must move software from one core to two. When the two cores have to communicate, you can lose a lot of performance. The system's latency gets worse but throughput stays high. That's what engineers must deal with when they break application code into pieces for each core. They have to balance latency against throughput."

But, engineers often can break code into pieces. "They might receive images through an Ethernet port, use an algorithm to process the images and then display them," explained Scidmore. "You can run a TCP/IP stacks on one processor, the algorithm on a second processor and dedicate a third processor to the display. If you have thousands of assembly-language files, you have no choice but to go back to block diagrams and start fresh with an eye toward dividing an application into separate threads."



Operating systems (OSs) that provide for symmetrical multiprocessing (SMP) may offer a way to integrate legacy and newer code. The Neutrino OS from QNX, for example, offers bound multiprocessing that lets developers dedicate, or "lock down," processes in a single core. "Suppose you have legacy software that might not behave properly in a multi-core chip or in a multi-processing environment," said Kerry Johnson, product manager at QNX. "The code might assume it has exclusive control of I/O devices, for example, which could cause problems in a multi-core system. In that case, you can 'lock' the software into a single core without modifying the code. Then you can add new application software that will take advantage of the remaining cores. The old and new software coexist under the same OS on one multi-core chip."

"It has been a challenge to get developers to understand that a multi-core chip does not require one OS per core," said Johnson. "A single OS with SMP capabilities readily oversees operations on many cores. As more OS vendors start to support SMP, developers will better understand they can use one programming style and have their code work on single-, dual- and quad-core processors. One set of software tools works in all those environments."

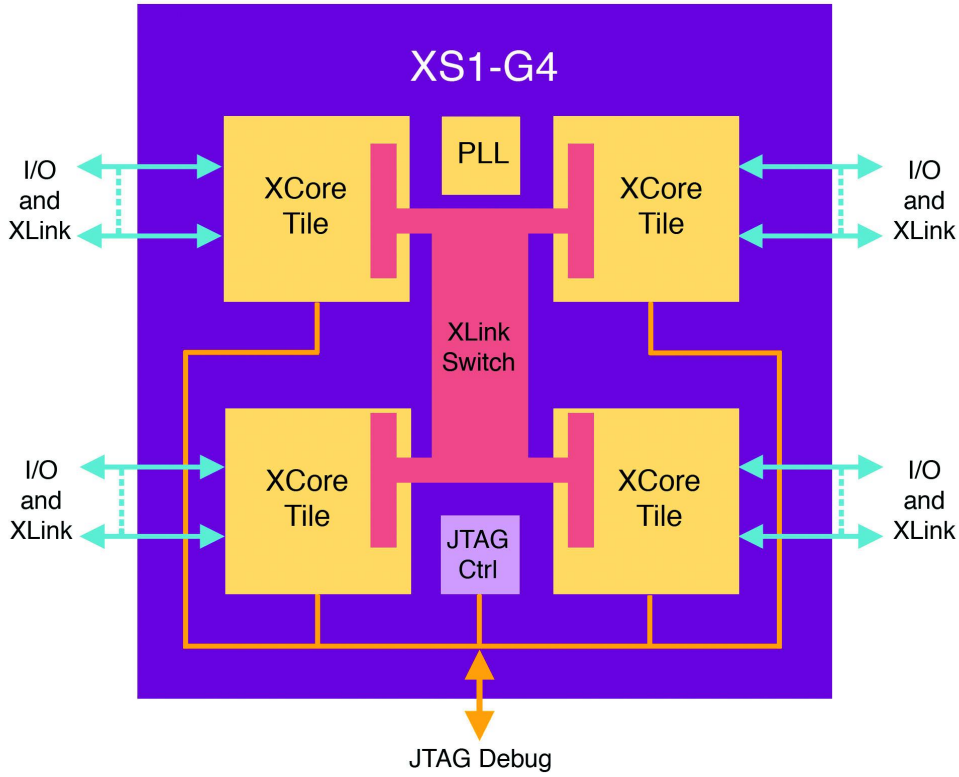


08ECN-Titus CS Multicore Art B

Those tools must include a source-level debugger. "You need one debugger that handles all the cores," explained Johnson. "You want to see everything in one window rather than have a debug window for each core. Regardless of which processor runs your code at a given time, the debugger must stop at break points, trace program execution and collect data across all cores just as it would for a single-core chip."

As part of its Neutrino operating system, QNX includes an instrumented kernel that lets developers log events such as state changes and interrupts at low levels. Thus you can see interaction across all cores. "When cores share memory and I/O devices, you could run into problems that you must examine in detail," said Johnson. "The instrumented kernel logs the data and our Momentics system profiler lets you analyze the logged information." QNX provides kernels with and without instrumentation. The former provides a way to examine and test operations in the field and the latter slightly decreases core overhead and slightly increases the kernel's code space.

Before you place application software into a multi-core system, you test it, often more times that you want to. But testing ties up expensive and difficult-to-deploy prototype hardware, if it even exists when you have code ready. And locking code to multi-core hardware too soon reduces opportunities to make hardware vs. software tradeoffs. As an alternative, consider running software on a virtual system that mimics your processor cores and other hardware devices. "Virtualization of your hardware lets you do a lot of exploring," said Michel Genard, VP of marketing at Virtutech. "You can see how application code works in one core or in several cores. And you can observe the effect of changing clock frequencies for each core, but without having to modify hardware. You don't tie up hardware and you can run your application code as much as you want."



Because a virtual system controls time, you can stop all the cores and device at once and you can run code backward or forward. You also can use checkpoints to access information at specific times from the 'internal' core and all other devices. "Developers tell us about nightmares such as SMP race conditions in multi-core systems," noted Genard. "They find it is almost impossible to debug multicore application code right on the hardware. On the other hand, a virtual environment lets them reproduce system behavior. They can 'back up,' inspect and change data or observe and change behavior as needed."

Virtualization has another benefit: Developers can determine how their software will utilize multiple cores and the system's other hardware. So they can make design decisions. They might find two cores will provide the same performance as four.

Engineers often look to Freescale, Intel, Broadcom, and others as vendors of multi-core chips, other companies, such as XMOS Semiconductor and Parallax also offer multicore devices, but aimed at different markets.

XMOS treats its XS1-G family of multicore ICs as "software defined silicon," according to David May, the company's chief technology officer and co-founder. "The XCores may seem unusual because they look like general-purpose processors as well as programmable state machines tightly integrated with I/O interfaces. They can do many things you would otherwise assign to dedicated hardware." Each XCore exists in an XCore Tile that has its own registers, memory and I/O ports as well as an XLink that communicates with other XCores on the same or separate ICs. An XCore can simultaneously execute eight threads.

Developers use the XC language, a derivative of C, which includes input and outputs commands and capabilities for the software components that will run in separate cores. "As much as possible, programmers have the same operators and data types as they find in ANSI C," said May. "Software developers are familiar with sequential C programs and they quickly grasp the new I/O operations."

The XC language's I/O commands control I/O ports and send messages between XCores via XLink channels. "An XLink connects cores on the same chip and we make external links available so you can connect two XMOS chips," said May. "In all likelihood products will use more than one of our ICs. In that case, you simply make the signal connections between the ICs. You don't need complicated high-speed buses or interfaces on your PCB. You don't have to worry about synchronous operations and operating systems. Our OS is in the hardware. Something that would be an OS call on most systems is an instruction or two on our processor."

Although a multi-core application might look formidable at the start, many tools exist to help developers take advantage of the available cores. The companies mentioned in this article provide a representative sample of ways to ease into the use of a multi-core chip in a product.

See the sidebars, "The 8-Core Parallax Processor Really Spins," **[1] and** "Virtual Platforms Speed Code Development for Multi-core SoCs" **[1]** here **[1]**

Links:
[1] http://www.ecnmag.com/cover-story-multicore-sidebar.aspx