



EE Times:

Multicore partitioning is a threads and comms problem

Richard A. Quinnell

(10/20/2008 12:01 AM EDT)

URL: <http://www.eetimes.com/showArticle.jhtml?articleID=211201707>

As more processor architectures turn to multicore designs, embedded developers are left with significant software development challenges. Tool providers and industry organizations have been making incremental progress toward addressing those challenges with their latest step simplifying software partitioning among cores. Vendors warn, however, that software developers must learn how to think outside the linear execution box.

Multicore processor architectures have been used in specialty applications such as cell phones for years. Now, however, general-purpose processors are turning to multicore architectures in droves. The traditional approach of speeding the clock to achieve faster execution has resulted in single-core processors consuming increasing amounts of power. To keep power use down while increasing performance, vendors have adopted the approach of integrating several, slower processors to do the work of the single fast one.

Multiprocessing software development can present some significant challenges, however, in both software design and debugging. Tool developers have chipped away at some of these challenges in the past few years. For instance, several tools, including Wind River's Workbench and ICE products, now support multicore debugging. Distributed operating system vendors, such as QNX Software, and multicore processor vendors, such as Texas Instruments and Toshiba, also offer debugging tools that help developers visualize and find problems when code segments running on different processors develop unexpected interactions. Compilers and OSes that support development of multithreaded software--only a step away from multiprocessing software--are also available. Even static analysis tools, such as those from Klocwork and GrammaTech, have leveraged multithreading techniques to tackle the multicore challenge, helping developers identify potential deadlocks and race conditions even before attempting to execute code.

One of the development challenges that tools have been hardest pressed to address, however, is shifting software from single-processor design to multicore. It's a two-part process, according to John Carbone, vice president of marketing for embedded RTOS vendor Express Logic. "To move software to multicore you first need to find a way to distribute the threads," said Carbone, "then enable communications among the threads across processor boundaries."

Partitioning for performance

The appropriate distribution or partitioning of threads among the processing elements is critical to use multiple cores effectively. Multicore hardware increases performance only to the extent that the cores remain busy. If the software is not partitioned properly, some cores will be overburdened and sluggish while others idle and waste CPU cycles. Worse, improper partitioning can increase contention among cores for shared resources, dragging down performance significantly.

The need for threads to communicate further complicates matters. It requires that developers incorporate mechanisms, such as message passing and semaphores, that software threads can use to exchange

information and coordinate actions across processor boundaries. Without careful implementation these mechanisms often end up configuration-specific, locking the software design into the partitioning and hardware choices made early in development. If analysis later reveals that additional cores or a redistribution of threads is needed, the communications mechanisms may require considerable revision.

To simplify implementation of these mechanisms, the Multicore Association, an industry group dedicated to addressing multicore design issues, recently released a Multicore Communications Applications Programming Interface (MCAPI) standard. Developers following the standard will be able to use tools to automatically create communications mechanisms--for example, PolyCore Software's Poly-Generator and Poly-Messenger/MCAPI, recently integrated into Express Logic's ThreadX RTOS.

The Poly-Generator tool allows developers to describe the characteristics and connection topology of their design's interprocessor communications channels using XML, according to PolyCore Software's president and CEO Sven Brehmer. The tool then generates C code and header files to create the necessary communications mechanisms that application code threads will use (see diagram, preceding page). By using the Poly-Messenger implementation of the MCAPI standard, Brehmer said, developers are freed from having to consider how the communications occur.

Because the underlying communications mechanisms stem from the system XML description and are abstracted from the applications programs, changing those mechanisms to accommodate new hardware or a repartitioning of software is easier. Simply alter the XML description and recompile, without changing anything else. The MCAPI standard, and the tools that support it, can thus help reduce the trial-and-test effort developers must put into finding the right software partitioning for their apps.

Think non-sequentially

Key to this effort, however, is writing the applications code to be multithreaded from the outset. This is still a challenge for many developers.

"It takes a long time for developers to get used to the consequences of threads interacting in multithreaded design on a single processor," concurred Mark Zarins, vice president of products at Gramma-Tech. "The consequences of thread interaction are worse in multiprocessing," he added.

This challenge has prompted at least one multicore processor vendor to take an entirely different approach: Ambric has its customers develop software for the company's massively parallel processor architecture (MPPA) using a structural object programming model. This method has developers describe their applications in block-diagram form comprising sets of software objects that link together. The company's tools then map each software object to its own processor in an MPPA array with hundreds of processors connected through self-synchronizing communications channels. Making the partitioning inherent in the software's structure eliminates the need to write code before partitioning.

Regardless of approach, developers must face the challenges of multicore development sooner or later. "This is a natural progression," said Brehmer.

Richard A. Quinnell (RichQuinnell@att.net) is a contributing technical editor based in Gardiner, Wash.