Mountain View, Calif. - Software's growing importance to system-on-chip design is no longer news; neither is the idea that today's SoCs typically comprise several processor types. But there has been surprisingly little discussion of what other steps those developments might enable.

When an embedded system included only a single processing site, and that site was an industry-standard microprocessor, real-time operating systems usually had the job of task management. The RTOS might be a commercial version or, more likely, an in-house product. In either case, its job was to schedule, dispatch, monitor and clean up after the various computing tasks and to give them a uniform, hardware-independent interface to the system resources.

That idea has proven itself, and there is evidence that the in-house RTOSes of the past are slowly giving way to small-footprint and low-royalty third-party products. But an RTOS generally hits a brick wall when it comes to multiprocessing SoC devices, especially when the various processing cores are of different architectures. There are very few RTOS products that can handle even symmetric multiprocessing, let alone heterogeneous piles of CPUs, DSPs and custom blocks.

What happens most often is that the RTOS on one centralized CPU communicates through semaphores, mailboxes, megaphones or anything else the architects can devise, with the ad-hoc control programs executing on the other sites. If some of those other blocks aren’t software-programmable, then a set-and-test register interface may have to suffice.

Everything becomes proprietary. Evaluation and performance analysis become, in a word, challenging. Managers get nervous. And people start asking questions about just how much of the software footprint, the CPU's execution cycle budget and the system-on-chip's energy consumption are going into the RTOS instead of the workload.

That is the situation that startup Ignios Ltd. is addressing. The Oxford, U.K., company has created a set of application programming interfaces for controlling the state of each task in a system, for interrupt handling and for intertask communication. The APIs are independent of the particular hardware blocks in the SoC, the tasks and the RTOS used in the system.

But that still leaves a great deal of hardware-dependent programming to do, to make sure control registers, interrupt handlers and task managers on each of the blocks in the chip communicate correctly with the APIs. And it still leaves a lot of time and energy going to task management.

So Ingios took a critical next step. The company created a block of hardware IP that handles task scheduling and intertask communications.

As described by vice president of business development Dan Chester, the block is something like an elaborate state machine that builds and traverses linked lists of task control blocks. Those task descriptors, as well as descriptors of the available hardware resources, are stored in a fast RAM
local to the block. The whole package, including the hardware IP, memory and API code, is known as SystemWeaver.

In operation, the SystemWeaver IP is instantiated on the bus, or perhaps a crossbar or other element of an SoC. Designers could run interrupt signals directly from individual blocks to the SystemWeaver block, but this is not strictly necessary.

Beyond insertion of the SystemWeaver block into the SoC design, each of the processing elements in the SoC may need the addition of a small bit of software or hardware. The overhead is necessary to signal the SystemWeaver block when a task has become blocked or is completed; Chester estimates it to require no more than about 10 kbits of code per block. In nonprogrammable blocks the function would be handled by a small state machine.

Finally, it is necessary to build the task and system resource structures in the SystemWeaver local memory.

Once the system is initialized, the SystemWeaver hardware acts like the task controller in an RTOS kernel, building a list of ready tasks and assigning a task to each hardware resource as that resource becomes available. The block also acts as the communications medium for inter-task communications.

**Task/data couplets**

Data sets are bound to task code, and when the task is ready, the couplet of task and data is dispatched to the first available appropriate processor. Prioritization and other quality-of-service dispatch schemes may be implemented.

If desired, a data set may be bound to two tasks that run on different processor blocks, so that when high-speed resources are tight, the task may be run on a slower, less critical resource. Other clever options are available as well, including a scheme for coping with run-time reconfigurable hardware resources.

When a task becomes blocked or completes, the processor that was executing the task signals the SystemWeaver block. The latter then traverses its lists and dispatches a new task to the processor in question.

It is possible, according to Chester, to initiate an entire third-party RTOS as a task, allowing it to run in the SystemWeaver environment, controlling its own tasks or simply handing off the task control responsibility to the SystemWeaver block. Ignios is courting relationships with RTOS vendors to develop a closer synergy between selected RTOS products and the system.

Functionally, the SystemWeaver block isn't doing anything that wouldn't go on inside an RTOS. But all of the critical paths for task dispatch are happening in hardware, slashing the intertask latency and removing what can be a considerable, top-priority load from the control CPU.

An additional benefit to the scheme is that the level of abstraction created by the SystemWeaver API is a natural place to conduct system-level debug. It allows the designer to examine the whole system at the task level, instead of delving into the source code for half a dozen different processors and trying to understand the state of a hardware state machine or two.

A comprehensive SystemC model of the hardware block is provided for system modeling, integration and verification.

SystemWeaver version 1.0 is now available for licensing to key customers. The package includes the
synthesizable RTL for the hardware block, the API code, the SystemC model and some reference design data on systems-on-chip using the block.