

Considerations in designing multicore systems

By Arun Subbarao



Multicore processing is having a tremendous impact on the development of embedded systems in the military and elsewhere. While previous multiprocessing solutions involved two or more physical chips, which doubled or more the amount of board space consumed, the introduction of multiple processing cores in a single chip allows operating systems and applications to leverage increased computing power. It has also provided access to additional computing resources without noticeably increasing the size or weight of the system.

But as developers delve deeper into the nuts and bolts of designing multicore systems and the applications that run on them, they're finding that these advantages come with certain obstacles and additional considerations that must be taken into account during the design process. The impact on software is most immediately felt in designing the OS, which must support either Asymmetric Multiprocessing (AMP) or, ideally, the more advanced Symmetric Multiprocessing (SMP) architecture. Concurrency and processor affinity issues also need to be addressed. For applications, there are considerations such as parallel execution and resource allocation.

Finding an OS to balance workloads

In multicore computing, the functions performed by the operating system become layered and more complex. The operating system design must be capable of handling the complex concurrency issues that arise with multicore architectures. Some of the generic areas of OS design affected by the presence of multiple cores are initialization, interrupt handling, scheduling, and locking. In the context of a Real-Time Operating System (RTOS), however, other key factors – such as priority scheduling, determinism, and interrupt latency – should be preserved in multicore architectures.

One such design optimization, known as *processor affinity*, may allow applications to request an affinity to a processor core. In this case, the operating system schedules the applications on the preferred processor core, as long as it does not affect overall system scheduling. A more rigid form of processor affinity is processor binding, where the task is always scheduled on the same processor core. This approach, however, may lead to priority inversions in RTOS environments. Operating system design should accommodate considerations such as processor affinity without degrading real-time determinism and responsiveness.

An RTOS such as LynxWorks' LynxOS supports SMP and can schedule tasks dynamically and transparently between processors to efficiently balance workloads using available processors. It optimizes load-balancing

support on multiple cores while preserving the key elements of real-time latency and determinism.

Optimizing applications for multicore architecture

Applications written for uniprocessor execution are not necessarily optimized for multicore architectures. Any inherent contention for resources that prevents execution parallelism could result in performance bottlenecks in multicore environments. Applications that are CPU-bound can exploit the full power of multicore architectures, while memory-bound or I/O-bound applications might need to be optimized to avoid the bottlenecks that arise due to bus contention in SMP architectures. Multithreaded applications can implicitly invoke resource contention as they request services from the operating system.

When looking at applications on multicore solutions, embedded developers must determine the allocation of functions within applications and redesign their applications to exploit parallelism. Developers must consider the design trade-offs of using multithreading versus nonmultithreading to harness the power of multiple processor cores. In some instances, applications may perform better on a single-core system.

Multicore computing continues to advance

New strides in multicore architecture make this an exciting time to be a developer. Every day, new approaches and strategies come to light that show us new ways to approach efficient multicore systems that are forming the heart of next-generation embedded computing systems. Choosing the most suitable OS for multicore architectures is a vital consideration, and overcoming issues like processor affinity, resource allocation, and parallel execution is also key to optimizing multicore systems. ✚

Arun Subbarao is the vice president of engineering at LynxWorks, responsible for LynxOS technology and product development. He has more than 11 years of software experience in the embedded industry working on UNIX, Linux, and RTOS kernels, networking protocols, and high availability. He received his bachelor's degree in computer science while studying in India, his master's degree from SUNY Albany, and an MBA from Santa Clara University. He can be reached at asubbarao@lnxw.com.