# Enabling Next-Generation Multicore Platforms in Embedded Applications



## The Challenge

With the recent advent of multicore technologies, multicore-based designs are being advocated for embedded systems. The reasons for this are twofold. First, multicore platforms offer greater computational capacity, with less space, cabling, and power consumption. Second, they offer the potential of replacing many interconnected processors with fewer multicore processors that are easier to manage and maintain. Of course, the main disadvantage of multicore designs is that software engineers now must explicitly deal with parallelism. In addition, consolidating different system components on a common platform requires mechanisms for temporally, logically, and securely isolating components, and such isolation is not straightforward to ensure. Such problems will become only worse as next-generation multicore systems become available, as such systems are expected to exhibit greater heterogeneity than current ones. In a *heterogeneous* platform, processing cores exist that have different functional or performance characteristics.

Greater heterogeneity will be driven by physical limitations that will eventually restrict per-chip core counts. When this happens, chip makers will distinguish themselves not by offering more or faster (general-purpose) cores, but by offering specialized hardware components that accelerate certain computations. Even now, heterogeneity exists at different levels in many commercial systems. One of the earliest notable examples of this was the IBM Cell Broadband Engine, which has a general-purpose CPU and eight special-purpose processing units on the same chip. More recently, many chip makers have been espousing *GPGPU* configurations as the new "standard computing platform"---in such a configuration, a general-purpose (GP) machine, with one or more cores, is paired with a sophisticated graphics processing unit (GPU), which may consist of many cores. Such configurations are part of a continuing evolution of co-processor-based designs; systems with digital signal processors (DSPs) or field-programmable gate arrays (FPGAs) as co-processors have been, and continue to be, in widespread use.

Unfortunately, heterogeneity makes multicore-related programming challenges even more difficult. Efficiently utilizing a multicore platform requires judicious resource allocation. Resource allocation problems are notoriously harder (and often provably so) when choices must be made involving resources with different capabilities.

## The Approach

In the real-time systems research community, prior work on heterogeneous multiprocessor systems has emphasized theoretical foundations. While such foundations are clearly important, given this emphasis on theory, we currently have very little understanding of how to *efficiently* manage heterogeneous multicore systems when supporting actual real-time workloads. In this project, we intend to re-examine and expand prior theoretical research on heterogeneous real-time computing; however, such efforts will be shaped by associated prototyping efforts that more carefully examine implementation issues and overheads. Such prototyping efforts will build upon prior implementation-focused efforts within our research group within the context of the LITMUS$^{RT}$ project.

Regarding heterogeneity, we intend to consider the use of both specialized, on-chip processors (as in Cell) and specialized off-chip processors (as in current GPGPU designs). We also intend to consider incidental heterogeneity that occurs due to high core counts: in systems with a large number of cores, NUMA-like behavior inevitably results, and this can be seen as a kind of heterogeneity, as code accessing data that is "close by" in the memory hierarchy runs faster than code accessing data that is "far away."

## Significance

The ultimate goal of this project is to provide a foundation for the design of next-generation real-time operating systems (RTOSs). Existing commercial RTOSs are rooted in uniprocessor resource-allocation principles devised decades ago and thus are completely ill-equipped to properly manage the complex multicore systems that are coming in the near future.

## Project Members

James Anderson, professor (PI)
Sanjoy Baruah, professor (PI)

## Research Sponsor

U.S. Air Force Research Laboratory, Rome, NY

## For More Information

Dr. James Anderson
Department of Computer Science
University of North Carolina at Chapel Hill
CB#3175, Frederick P. Brooks, Jr. Building
Chapel Hill, NC 27599-3175
Phone: (919) 962-1757
Fax: (919) 962-1799
E-mail: anderson@cs.unc.edu

**http://www.cs.unc.edu/~anderson/projects/nextgen.html**