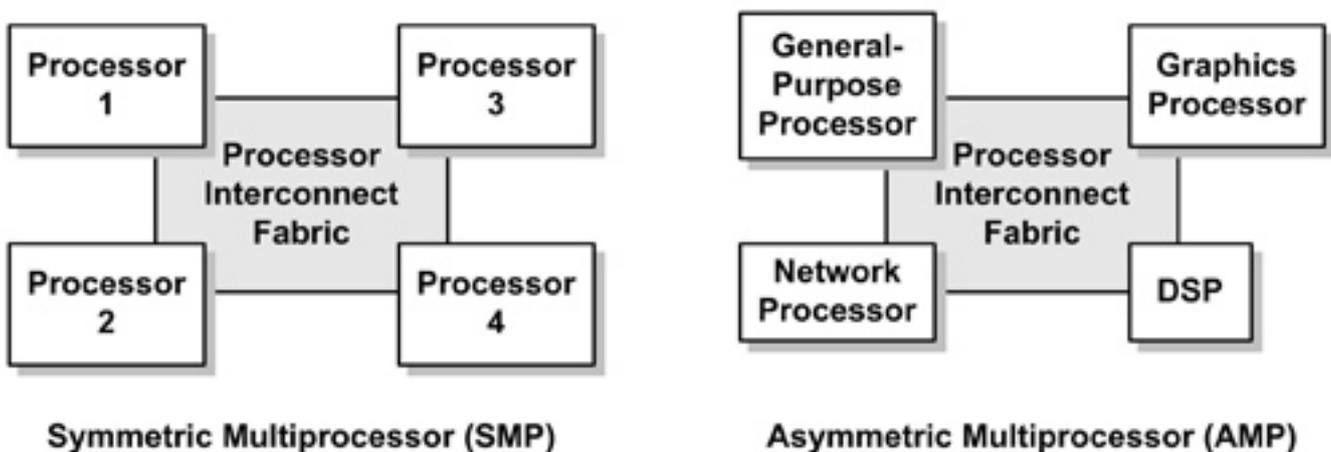


## How to Survive the Quest for a Useful Multicore Benchmark

Shay Gal-On, Markus Levy, and Steve Leibson

Selecting an optimal multicore chip or SoC architecture is considerably more complex than picking a lone processor chip or core. As with processor selection, benchmarks can help. A good multicore benchmark will identify bottlenecks in the multicore system design including memory and I/O bottlenecks, computational bottlenecks, and real-time bottlenecks. In addition, a good multicore benchmark will identify synchronization problems where code and data blocks are split, distributed to various compute engines for processing, and then the results are reassembled. There's an art to synchronization and multicore system designs can either help or hinder this important facet of multicore processing. The best multicore benchmark is your application code, but that's not always available. If available, it still may not be practical to port the code solely for experimental purposes. In fact, application code is frequently not available when a multicore hardware architecture must be chosen unless you are migrating existing application code to a multicore implementation. When actual application code is not available, you can use task surrogates as stand-ins for the big chunks of your application to test the mettle of a multicore chip or proposed SoC architecture.

Many people think that a specific processor vendor invented the multicore chip. While the x86 architecture gets a hugely disproportionate amount of publicity, multicore processors based on this architecture represent only a small fraction of the multicore chips shipped each year. Even if you only consider symmetric multiprocessing (SMP) chips, the number of multicore vendors is large and includes AMD, ARM, Freescale Semiconductor, IBM, Intel, and MIPS Technologies, among others. Many more chip vendors build multicore ASSPs with heterogeneous architectures. Figure 1 compares SMP and heterogeneous architectures.



**Figure 1: Comparison of SMP and Heterogeneous Multicore Architectures**

Multicore ASSPs and similarly architected SoCs can be simple 2-core designs that combine a general-purpose processor and a DSP core, or complex manycore devices that incorporate tens or hundreds of processor cores. Multicore chips employ a variety of interconnect technologies including buses, point-to-point connections, and NoCs (networks on chip). Consequently, multicore architectures are highly differentiated and multicore benchmarks must be differentiated as well.

First- and second-generation processor benchmarks for single-core processors are well-established, well-known, and familiar so processor vendors, system developers, and academic research groups prefer to use them to measure multicore processor performance. In theory, it's possible to do this by launching multiple instantiations of each benchmark. SPECrate (from SPEC, the Standard Performance Evaluation Corporation) uses this method to measure a system's ability to process jobs of a specified type in a given amount of time. SPEC notes that this "metric is used the same for multi-processor systems and for uni-processors." EEMBC also provides the ability to launch multiple instantiations of its CoreMark benchmark (<http://coremark.org/blog.php?pg=blog&link=/wordpress?p=48> [1]), but this benchmark, used in this manner, certainly doesn't reflect a real-world application.

In reality, it's not possible to guarantee that a multicore system is exercising more than one processor core unless you employ some form of processor affinity. Without programmer intervention, a multicore platform's scheduler assumes control over execution of the individual benchmark instantiations and without control over processor-task assignments, the benchmark is severely compromised as an accurate measuring tool.

### **Multicore Benchmark Criteria**

Can you use any sequential code for this purpose? To answer this question, you need to understand that benchmarks for multicore architectures should be either computationally or memory intensive, or some combination of both. Whether SMP or AMP, a multicore chip's memory bandwidth plays a key performance role, and memory bandwidth depends on the multicore architecture, the interconnect fabric's bandwidth under changing loads, and the memory subsystem's design.

Shared memory, controlled by some type of locking mechanism to prevent simultaneous access by multiple cores, provides a straightforward programming model because each processor can directly access the same memory. Shared memory is also relatively slow because it's a shared resource and locks impose overhead. Typically associated with homogeneous SMP systems, shared memory facilitates programming with traditional languages because it allows passing data by reference, without actually moving the data.

Distributed memory systems, which are more commonly used for SoC designs, give each processor its own local memory. When one core requires data from another core or when processor cores must synchronize data, the data must be moved from one local memory to the other or the control code must switch from one processor core to the other (which is nearly impossible for heterogeneous systems). Even when each processor core has its own local memory, there can still be memory

bottlenecks depending on how data moves around and onto or off of the chip. A good multicore benchmark will detect such memory bottlenecks.

Scalability is another important benchmark criterion. It's not uncommon to have hundreds of threads in a relatively complex program. If the number of threads exactly matches the number of processor cores, performance could scale linearly assuming no memory-bandwidth limitations. Realistically the number of threads often exceeds the number of cores and performance depends on other factors such as cache utilization, memory and I/O bandwidth, inter-core communications, OS scheduling support, and synchronization efficiency. A processor incurs performance penalties when it oversubscribes computing resources and a good multicore benchmark will highlight such situations.

### **MultiBench: A Multicore benchmark**

A benchmark that executes multiple copies of sequential code doesn't probe one of the most important multicore benefits: using parallelism to improve the performance of individual tasks rather than overall system throughput. Multicore benchmarks that employ task decomposition, functional decomposition, and data decomposition can measure this characteristic of multicore systems.

EEMBC's industry-standard MultiBench tests are oriented toward general-purpose processors based on SMP architectures. MultiBench extends the scope of sequential EEMBC benchmarks so that they can be used to evaluate multicore architectures, identify memory bottlenecks, explore OS scheduling support, and test synchronization efficiency (see Figure 2). Although MultiBench implements significantly more parallelism than the consortium's first- and second-generation benchmarks, it's not a benchmark for heterogeneous multicore architectures.

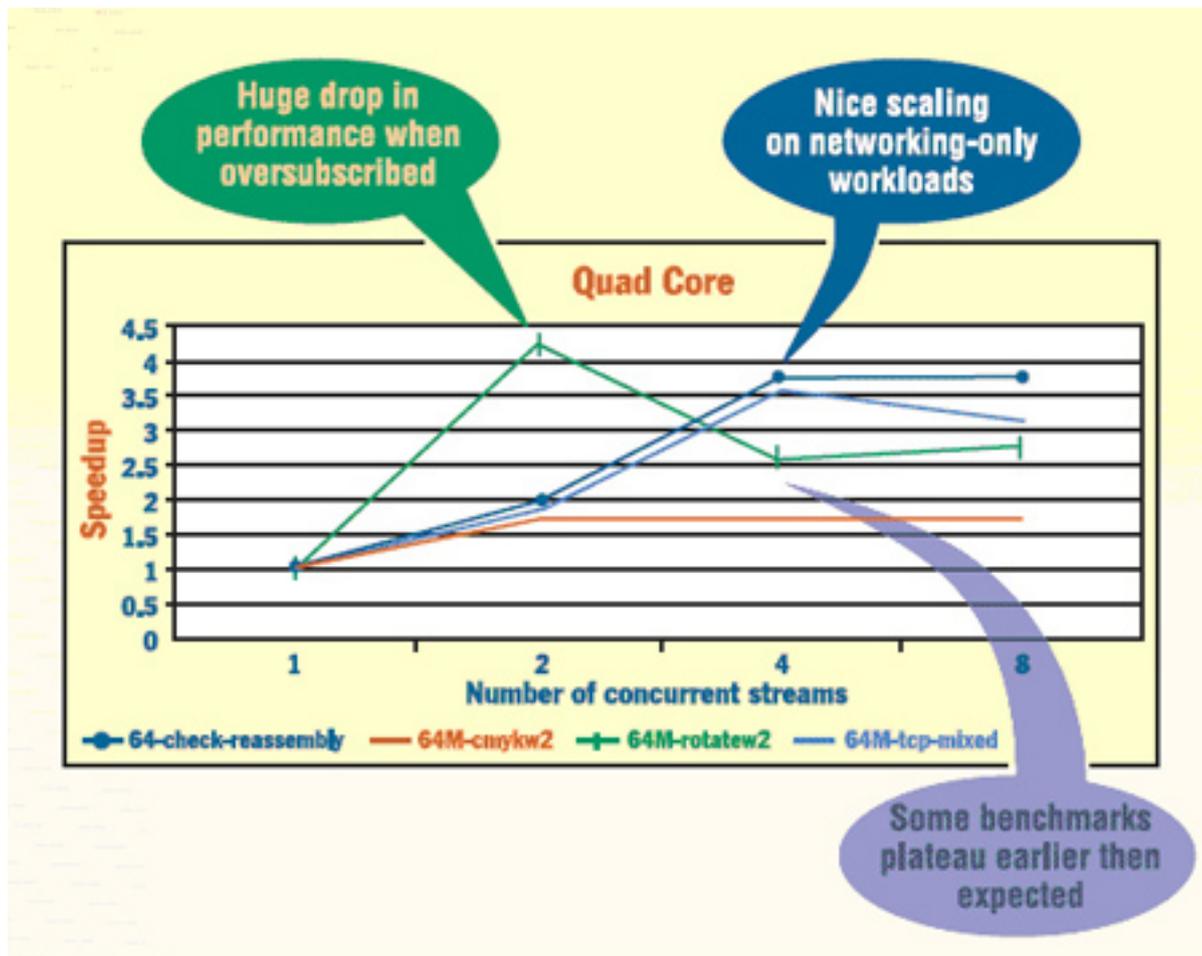


Figure 2: EEMBC's MultiBench identifies many multicore bottlenecks

Heterogeneous multicore architectures require an entirely different benchmarking strategy. Unlike the SMP benchmarks, heterogeneous cores require careful workload partitioning because that's exactly the way they're used in practice. Benchmarking heterogeneous devices in a relevant manner with portable code is a huge challenge because each part of the system likely uses a different development-tool chain and communication among the different processor cores isn't standardized.

Extending the existing MultiBench framework to support heterogeneous systems requires a portability standard. One such standard is the Multicore Association's Multicore Communications API (MCAPI), which provides a standardized framework for partitioning benchmark code into intercommunicating blocks.

## Application-specific standard Benchmarks

The demand for SMP and heterogeneous multicore benchmarks is growing almost exponentially and a move to "scenario-oriented" or application-specific standard benchmarks (ASSBs) is afoot. ASSBs are black-box benchmarks: they specify only the input, expected output, and interface points. In other words, it doesn't matter what's inside the system being benchmarked as long as it delivers the expected output.

## How to Survive the Quest for a Useful Multicore Benchmark

Published on Electronic Component News (<http://www.ecnmag.com>)

---

Although ASSBs have been described as the way forward in computer benchmarking, they don't come without challenges. First, creating an industry standard, at least within the EEMBC domain, requires a consensus among the members to select the right mix of test scenarios from among an almost infinite number of possibilities. Second, getting an ASSB to run properly on an evaluation platform requires the careful assembly of many system-level components, including both hardware and software, to get consistent results.

SMP or heterogeneous, measuring multicore performance requires new benchmarks. Coming up with these benchmarks is only half of the challenge; the other half is interpreting the results. That's where the fun really begins.

### **Source URL (retrieved on 08/29/2014 - 10:13pm):**

[http://www.ecnmag.com/articles/2009/12/how-survive-quest-useful-multicore-benchmark?qt-recent\\_content=0](http://www.ecnmag.com/articles/2009/12/how-survive-quest-useful-multicore-benchmark?qt-recent_content=0)

### **Links:**

[1] <http://coremark.org/blog.php?pg=blog&link=/wordpress?p=48>