How to Choose a CPU Core for Multi-CPU SOC Designs

MIPS Technologies, Inc.
June 2002

The use of multiple CPUs in SOC designs is becoming increasingly popular. Processor cores being considered for multi-CPU designs should possess several important features, including excellent performance density, efficient inter-processor communications, debug support, and implementation flexibility and configurability. This article examines these features in relation to a core specifically designed for multi-CPU designs, the MIPS32™ M4K™ core.

Due to an increasing demand for programmable performance, the use of multiple CPUs is becoming very popular among SOC designers. For many applications, performance requirements are increasing faster than the ability of a single CPU to keep pace. The allocation of performance, and thus response time, for complex real-time systems is often easier with multiple CPUs. And dedicated CPUs in peripherals or special accelerators can offload low-level functionality from a main CPU, allowing it to focus on higher-level functions.

Multi-CPU designs are feasible in embedded systems today for various reasons. Embedded applications often have a great deal of partitioning flexibility, which facilitates mapping to multiple CPUs. In some cases, it is much easier to map to multiple separate CPUs than to try to use a single one. Similarly, many embedded applications have obvious parallelism that can be exploited with parallel CPUs. The NPU designers, for instance, have taken advantage of this for routing. And finally, with today's 0.13- and 0.10-micron processes, multi-CPU SOC systems can be built economically.

Multi-CPU designs can be found in many important growing markets. Network routing is one of the leading applications where multi-CPU designs have become popular. Most of the new generation of network processors are based on multi-CPU designs. In addition to these standard NPUs, targeted ASSPs are using multiple processors to build more optimized, application targeted routing solutions. For many, a targeted ASSP is a much more efficient solution than a standard NPU. An ASSP can integrate the right physical interfaces or target the on-chip memory size or provide the best configuration of processors.

Related applications such as DSLAMs and base stations, or high-performance networked storage devices, also work well with a divide-and-conquer approach to building high-performance programmable solutions. Even end-user equipment is beginning to utilize multiple CPUs to get

the best performance density for cost and/or power reasons. Set-top boxes, residential gateways and even smart mobile devices are being built with multiple CPUs — and not just one RISC processor and one DSP, but multiple quantities of each.

What Makes a Good CPU Core

For a processor core to be a good candidate for multi-CPU designs it must possess a few important features.

First and foremost, it must provide excellent performance density. The goal in a multi-CPU design is to attain as much aggregate performance as possible per square millimeter or per watt. The solution that has the most MIPS in the smallest space or the lowest power is preferred.

Another requirement is efficient inter-processor communications. Even though many multi-CPU designs are software coherent, there is still a significant amount of inter-processor communication. If a framework for the support of communication between the processors is not provided, then it will have to be developed by the SOC designer. This is not only time-consuming; it also introduces another level of complexity to the design. Providing support for inter-processor communication simplifies the SOC designer's task and shortens development time. It can also minimize inefficiencies at the partitioning boundaries.

Multi-CPU designs can be a challenge to debug. The cores interact with and are dependent upon one another, and it's important to see this interaction when debugging an SOC. This requires that the CPUs have a built-in capability that allows them to be debugged at the same time while also fully interacting. Without this, SOC debug can quickly become a nightmare or, worse, it may be impossible to fully debug. It is also critical that the CPUs be supported with good multi-CPU debug tools that allow the SOC designer to take advantage of the debug capabilities built into the CPUs.

Something else to consider is the level of flexibility in configuring and implementing the CPU core. In a multi-CPU design, it is important to minimize area and power consumption while maximizing performance, and a highly configurable, synthesizable CPU enables the designer to target area, power and frequency for the specific application. This is not possible with a hardened core, or one that isn't configurable, which severely limits the implementation options.

With these factors in mind, MIPS Technologies recently launched a CPU core specifically designed for multi-CPU designs. The MIPS32™ M4K™ core is high-performance, yet, perhaps surprising to some, is also small and low-power. It is also synthesizable and has a low-latency memory system.

Let's examine how the M4K core maps to the requirements listed above.

Performance Density

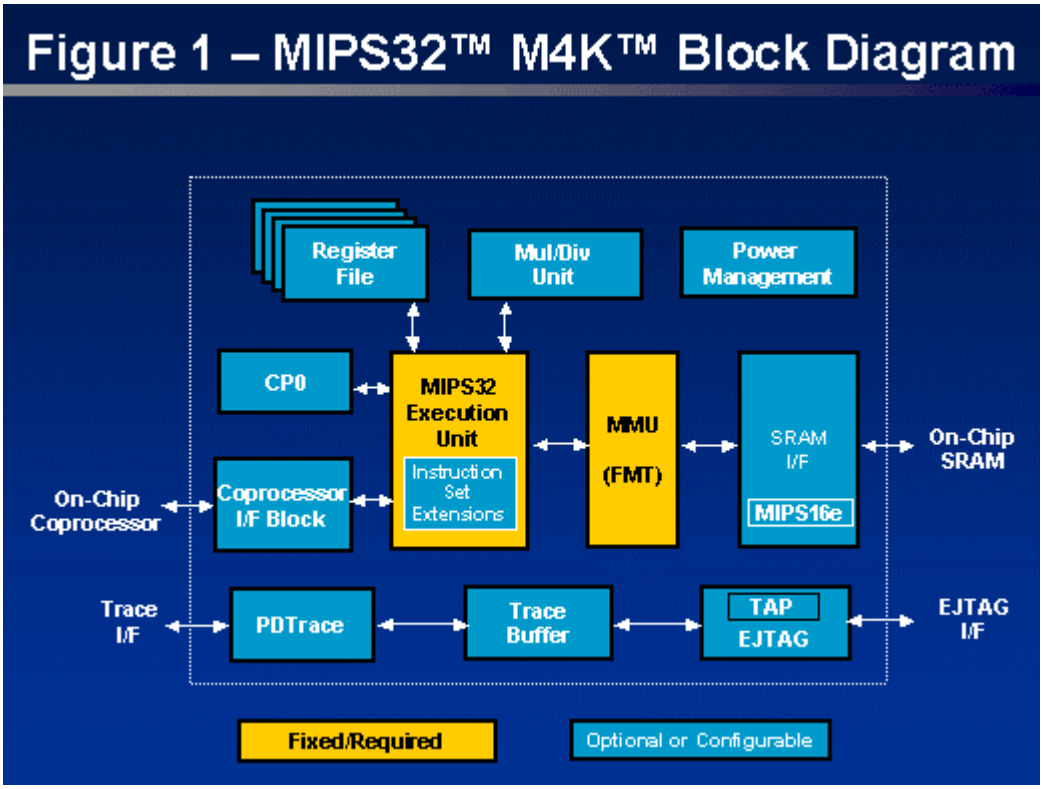
In terms of performance density -- the best performance in the smallest space or with the lowest power consumption -- the M4K core provides high performance within tight area and power constraints. It delivers approximately 1.35 Dhrystone MIPS/MHz (without using questionable compiler tricks that might be found with other cores), and it runs at up to 240 MHz (worst case)

in a generic 0.13-micron process, or up to 300 MHz in a higher performance 0.13-micron process.

On the denominator side of the performance density equation, the M4K core is an extremely small, very low-power processor. It can be configured to be as small as 32K gates or less than 0.3 mm^2, while still maintaining full MIPS32 architecture compatibility. Its power consumption is extremely low, as low as 0.10 mW/MHz in a generic 0.13-micron process.

The core also includes new MIPS® architecture enhancements to improve performance for important application functions. They include prioritized, vectored interrupts and up to four register contexts to minimize interrupt latency and overhead. Bit-field and byte-level instructions are included to provide highly efficient packet handling.

User-defined instruction-set extensions have been included for SOC designers who want to implement proprietary, highly focused application optimizations and extend the industry-standard MIPS32 instruction set in the M4K with custom instructions. Any register-to-register or register-immediate instruction can be added. The custom instructions can be single or multi-cycle and new user state is also supported. These user-defined instructions are supported in the core RTL, synthesis scripts and simulation models, and can be used with industry-standard development tools from companies such as Green Hills, Cygnus and Mentor.



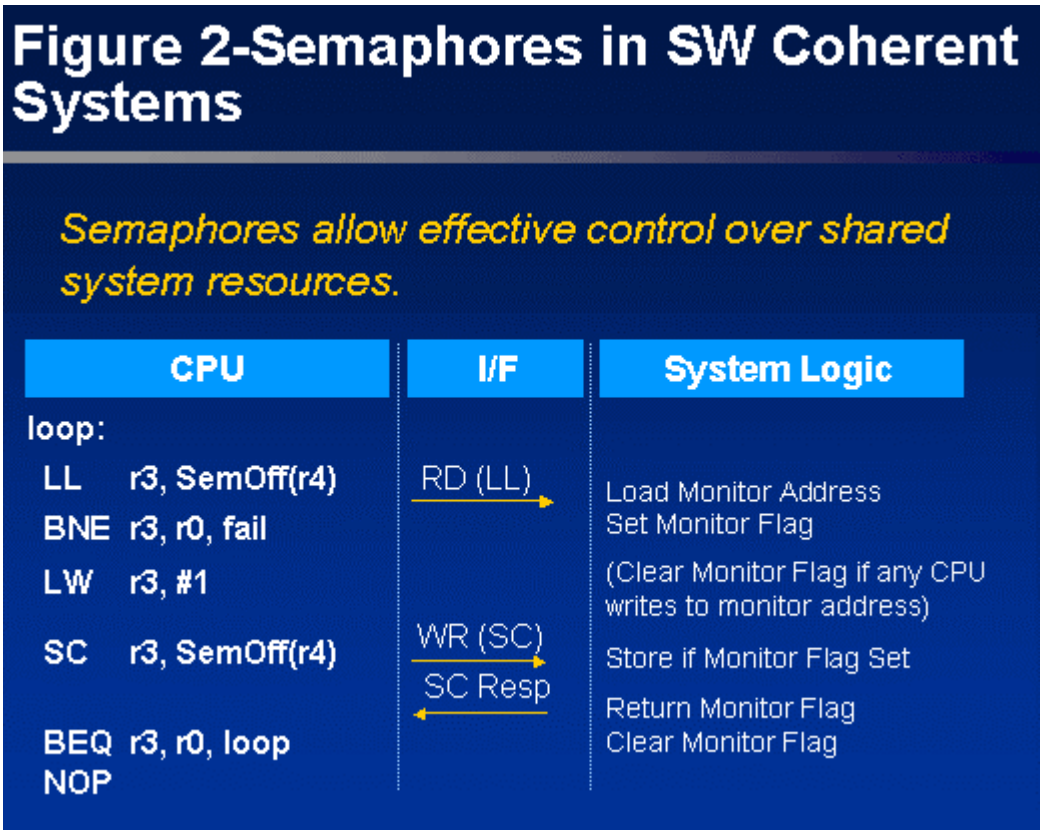
In addition – and perhaps most importantly – the M4K core features all of this while maintaining complete MIPS32 compatibility and working with the wide range of software and tools available for MIPS CPUs. This core is, in fact, the only available configurable processor with custom instruction extensions, based on a well-supported, industry-standard architecture.

Inter-Processor Communication

In multi-CPU designs, the processors have to communicate with each other to control access to shared resources. In most of the multi-CPU designs being built today, memory coherence is managed completely in software, so the normal cache coherency mechanisms for communication are not available.

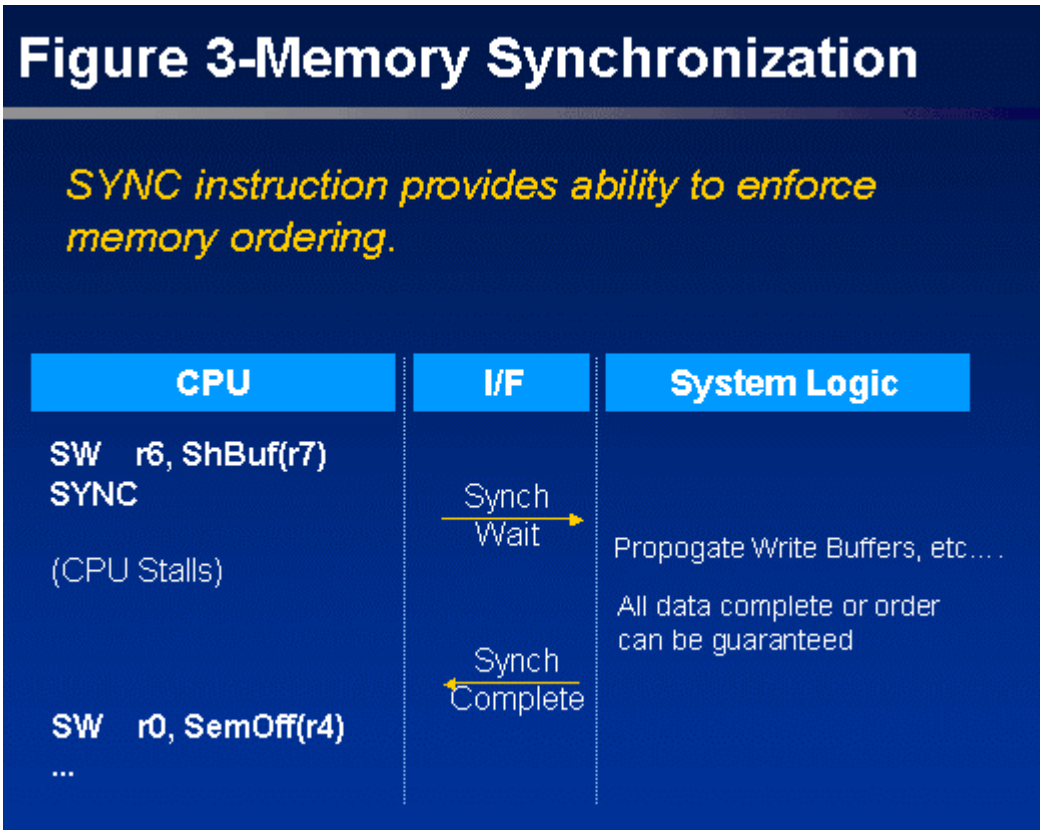
The M4K core includes support for multi-CPU semaphores by externalizing the behavior of the Load Linked (LL) and the Store Conditional (SC) instructions in the MIPS architecture to allow the system logic to maintain the necessary monitoring. In addition, the behavior of the SYNC instruction, which provides a memory ordering barrier to ensure correct ordering semantics, is also brought out to the signal interface to give the system design maximum control and flexibility.

As shown in Figure 2, when a typical semaphore access code sequence is executed, the LL and SC instructions provide the behavior of an atomic Read Modify Write sequence without the need to propagate a lock through the system.



For example, in a test-and-set spin-lock, an LL instruction generates a read on the signal interface that is identified as an LL to the system. The system then sets a monitor on that address. If another processor writes to the monitored address, the monitor's pass/fail flag is cleared. When the CPU executes the SC instruction, the system logic conditionally executes the store to memory depending on the state of the monitor. The pass/fail indication is returned to the processor, where the software typically repeats a failed sequence until it succeeds.

Similarly, the semantics of the SYNC instruction are externalized to allow the system to ensure the correct behavior of the memory barrier.



As shown in the example in Figure 3, a SYNC is placed between the last store to a shared buffer and the store that releases a controlling semaphore. When the SYNC instruction is executed, the CPU signals to the system logic that the CPU is waiting for a synch response. Once the system is sure that ordering can be guaranteed, it signals it back to the CPU, allowing subsequent loads or stores to issue. A simple system may wait for all outstanding transactions to complete; or, a more complex system that can maintain order in its queues and buffers may respond more quickly. Either way, the CPU software and hardware mechanism is the same.

The inter-processor communications features on the M4K are an elegant solution, fully supported by industry-standard development tools. A designer implementing a multi-CPU design

using the M4K core can use these features to quickly and easily communicate between cores, reducing design time and lowering implementation risk.

Debug Support

Debugging a multi-CPU design can be a challenge. The interactions between multiple processors can create scenarios that are difficult or impossible to identify and resolve unless debugging can be performed on all of the cores at the same time.

The M4K core has features in its EJTAG-based debug logic to ease this challenge. First, the EJTAG interface on the M4K core can be daisy-chained with all of the cores on an SOC to allow a single debug probe and host to debug all of the cores simultaneously. This also enables support of independent or synchronous start, stop and single step control.

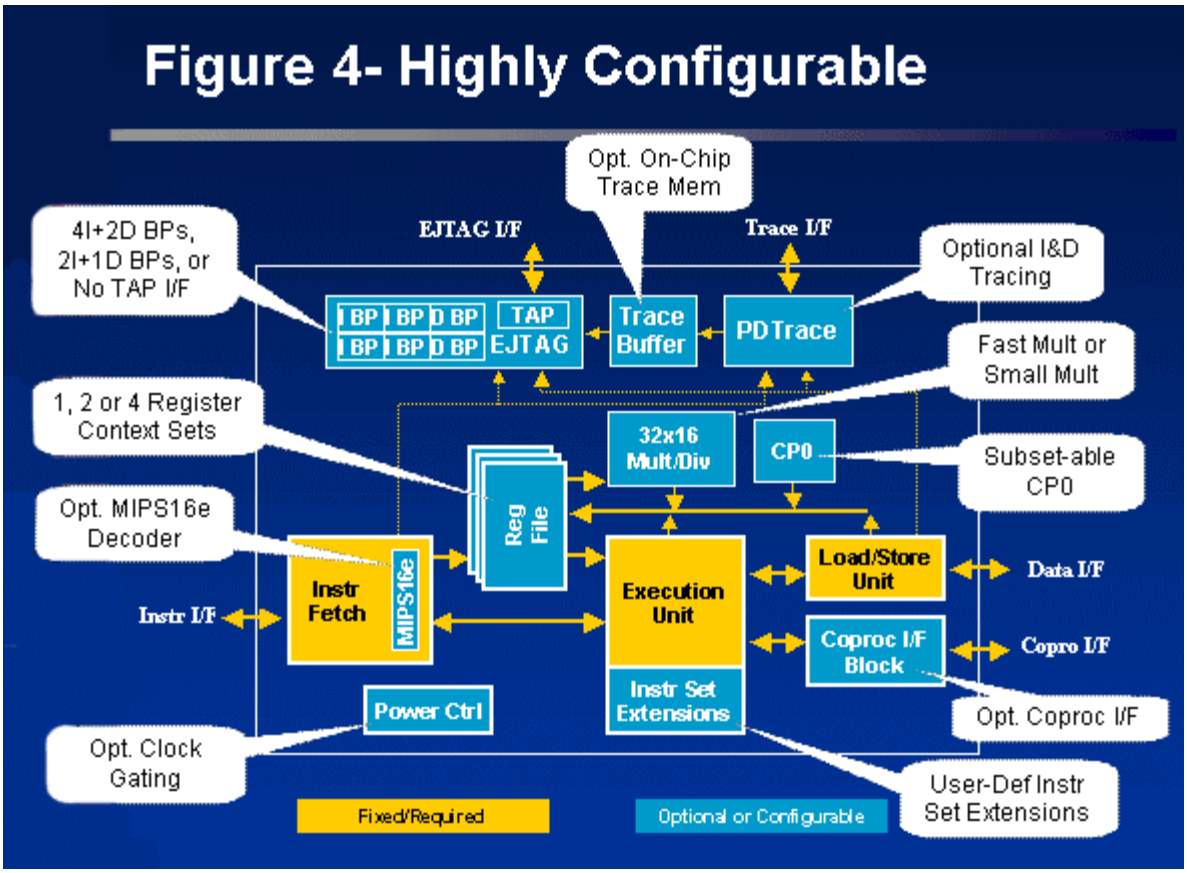
Another nice feature for multi-CPU debug supported on the M4K core is cross CPU breakpoints. The core supports the ability for a breakpoint in one CPU to force a breakpoint in one or more of the other CPUs within a few clocks. This is software controlled through a small logic block that defines which CPU can cross break any other CPU. This capability is very useful for debugging problems that involve CPU interactions.

These debug features on the M4K core reduce design time and risk by offering the user a fully supported, easy-to-use debug environment for multi-CPU SOC designs.

Configurability

Since efficiency is critical in multi-CPU systems, implementation flexibility and configurability are crucial to building the optimal solution for the particular application problem being solved.

The M4K core is the most configurable core MIPS Technologies has ever developed. As shown in Figure 4, most of the CPU blocks are optional or configurable.



The multiplier can be targeted for high performance or for minimal area. As mentioned previously, the number of register contexts can be configured. The MIPS16e™ code compression decoder can be removed if not needed. Multiple tradeoffs in debug support can be made with either minimal support, and different numbers of hardware breakpoints can be defined. Even the TAP controller is optional.

For program and data trace, an on-chip capture buffer can be included or excluded, and trace support can be configured out. The coprocessor interface logic can be configured in or out. User-defined custom instruction extensions can be added, as mentioned previously, and clock gating (to reduce power consumption) can be included or excluded.

In addition to this internal core configurability, the M4K core supports considerable flexibility in the memory system design. It can support Harvard-type systems with separate instruction and data memory spaces using a very low-latency synchronous SRAM style interface for highly efficient memory access. This interface supports single-cycle and multi-cycle transactions and supports connecting 8-, 16-, or 32-bit peripherals. Or, it can be built for a shared memory system with a combined instruction and data space using the same low latency interface.

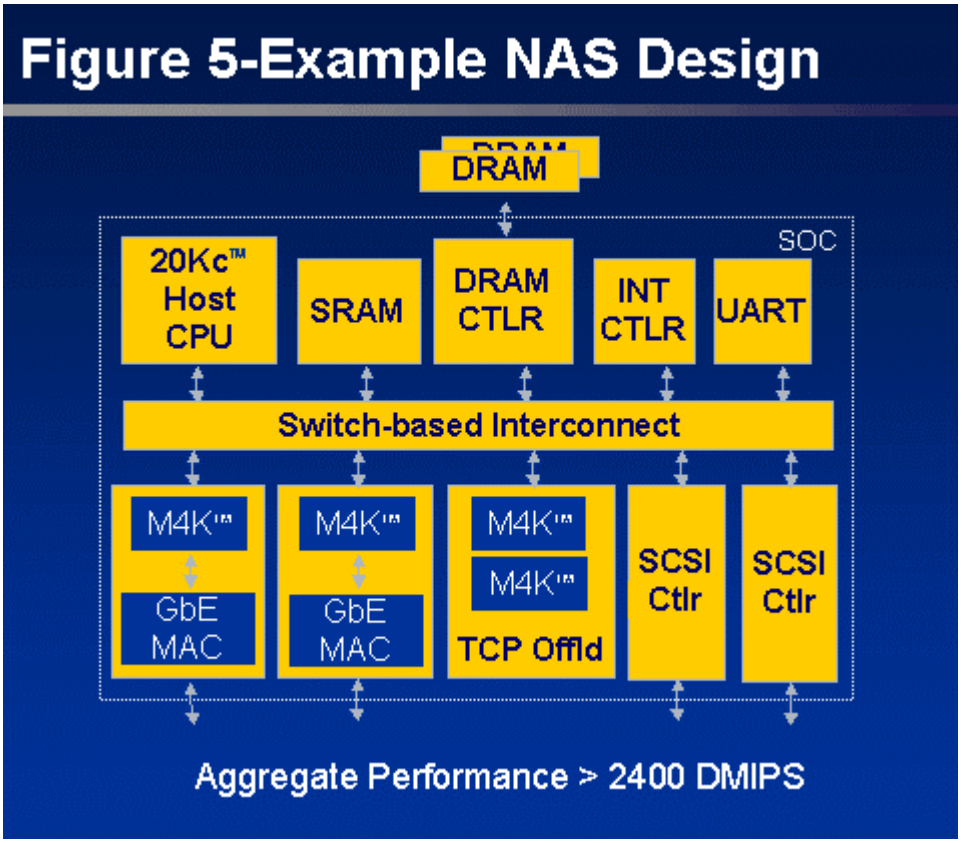
In addition, a bridge from the SRAM-style interface to EC-based system logic supports use of existing peripherals built for systems based on MIPS32 4K™, 4KE™, or MIPS64™ 5K™ cores.

This bridge supports low latency memory to maintain high-performance access to local memory while connecting to a traditional MIPS-based™ CPU subsystem.

Multi-CPU Systems

What can systems do with multiple CPU cores? As shown in Figure 5, a network storage system can use multiple cores to offload specialized functions from a host processor, a MIPS64 20Kc™ core, for instance. By putting a CPU at network interfaces, higher level functions like filtering, L2 or L3 protocol response and segmentation and reassembly can be done locally by the peripheral device, freeing up the host CPU for higher level protocol and management functions.

Similarly, using one or more M4K cores in an accelerator, like the TCP offload in this example, allows the partitioning of special high-performance functions away from the host processor, while still maintaining the advantages of a standard reprogrammable device.



Another common example of a multi-CPU system design is a line card. This may be for a network router, a DSLAM, or a wireless base station. In these cases, parallel CPU subsystems, sometimes called micro-engines, are used to provide a tremendous amount of aggregate performance for highly parallel applications, such as level 2 processing, packet classification, filtering, or tag management. In this case, the host processor is used primarily for exception processing. There are many variations of this approach using not only parallel processing but also pipelines of processors or pipelines of parallel processors.

Multi-CPU design is becoming increasingly popular with chip designers because it provides scalable, programmable performance and fits the natural partitioning of many networking and other embedded system designs. As with most SOC design today, multi-CPU design is not without challenges, but those challenges are lessened considerably when using a CPU core like the MIPS32 M4K that is optimized to handle the job.

David Courtight is director of product strategy at MIPS Technologies, Inc.