United Business Media

Learn today. Design tomorrow.

ESD

Embedded Systems Design

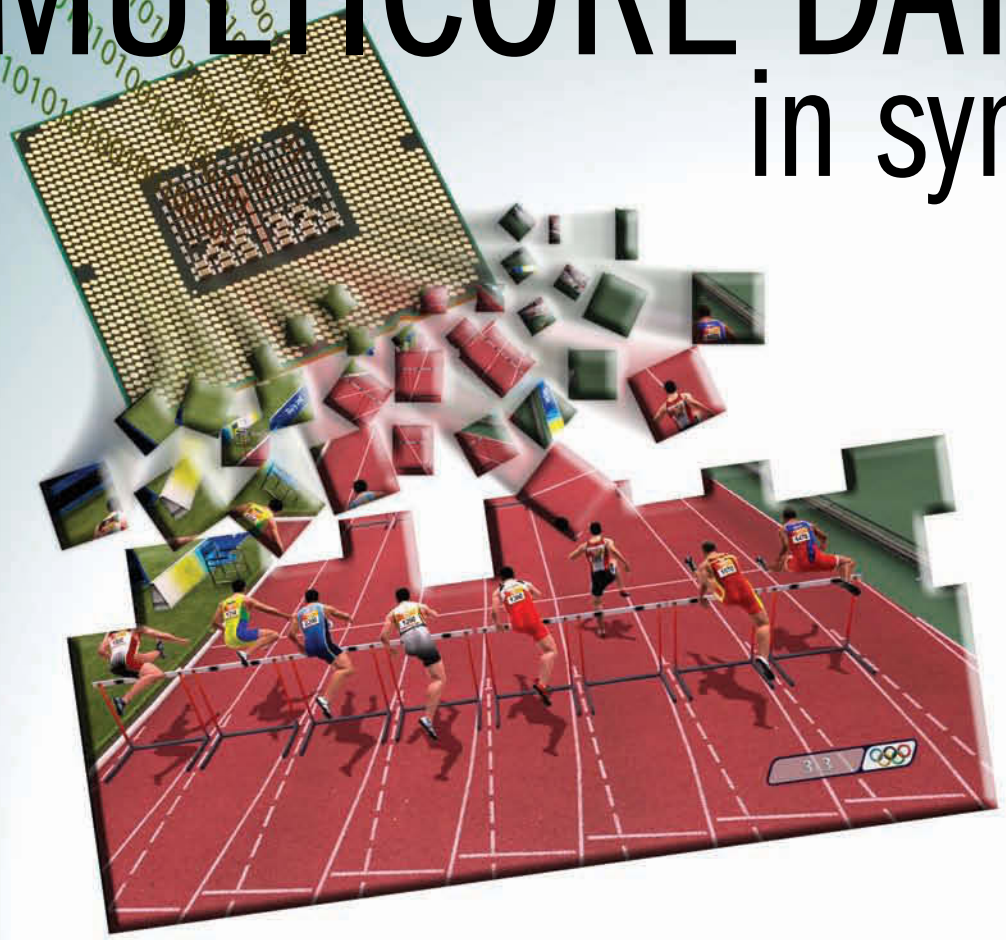
The Official Publication of The Embedded Systems Conferences and Embedded.com

VOLUME 22, NUMBER 9

OCTOBER 2009

Keep your MULTICORE DATA in sync

16

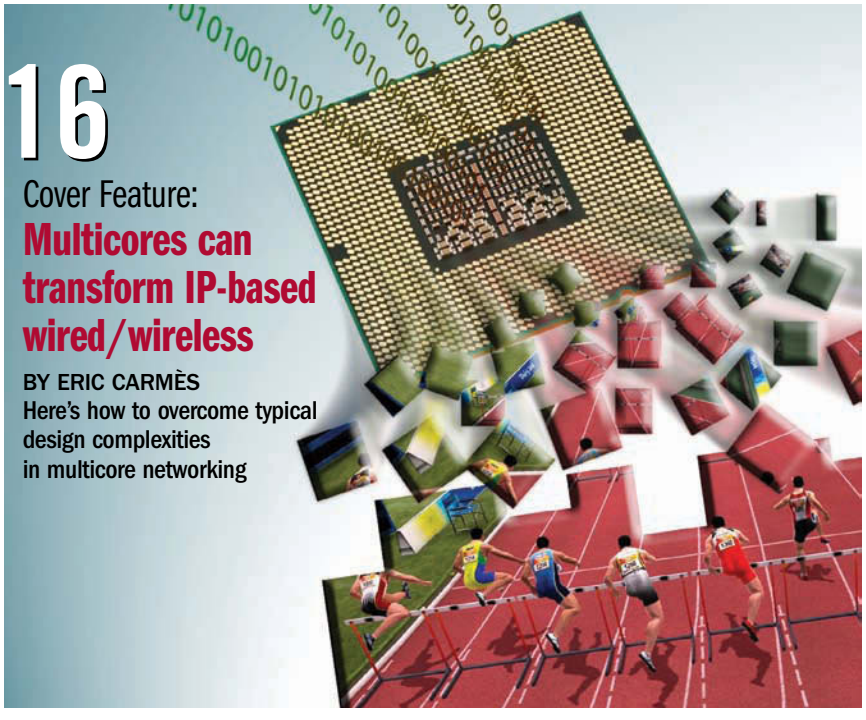


Learn today. Design tomorrow.

ESD

EMBEDDED SYSTEMS DESIGN

VOLUME 22, NUMBER 9
OCTOBER 2009



16

Cover Feature:

Multicores can transform IP-based wired/wireless

BY ERIC CARMÈS

Here's how to overcome typical design complexities in multicore networking

25

Multicore programming made easy?

BY RICHARD STAHL

IMEC's MPA tool transforms sequential C into parallel code.

33

Charting the new MCUs

BY STEVE BITTON

Here's a quick look at some of the current microcontrollers on the market for embedded systems.

COLUMNS

DEPARTMENTS

IN PERSON

ESC Boston
September 21–24, 2009
www.embedded.com/esc/boston

ESC UK
October 6–8, 2009
www.embedded.co.uk

ESC Grenoble
December 1–3, 2009
www.design-reuse.com/ipesc09/

ESC Silicon Valley
April 26–29, 2010
www.embedded.com/esc/sv

ONLINE

www.embedded.com

EMBEDDED SYSTEMS DESIGN (ISSN 1558-2493 print; ISSN 1558-2507 PDF-electronic) is published monthly with the exception of a combined July/August issue by TechInsights, 600 Harrison Street, 5th floor, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. SUBSCRIPTION RATE for the United States is $55 for 12 issues. Canadian/Mexican orders must be accompanied by payment in U.S. funds with additional postage of $6 per year. All other foreign subscriptions must be prepaid in U.S. funds with additional postage of $15 per year for surface mail and $40 per year for airmail. POSTMASTER: Send all changes to EMBEDDED SYSTEMS DESIGN, P.O. Box 3404, Northbrook, IL 60065-9468. For customer service, telephone toll-free (877) 676-9745. Please allow four to six weeks for change of address to take effect. Periodicals postage paid at San Francisco, CA and additional mailing offices. EMBEDDED SYSTEMS DESIGN is a registered trademark owned by the parent company, TechInsights.

Here's how to overcome typical design complexities in multicore networking.

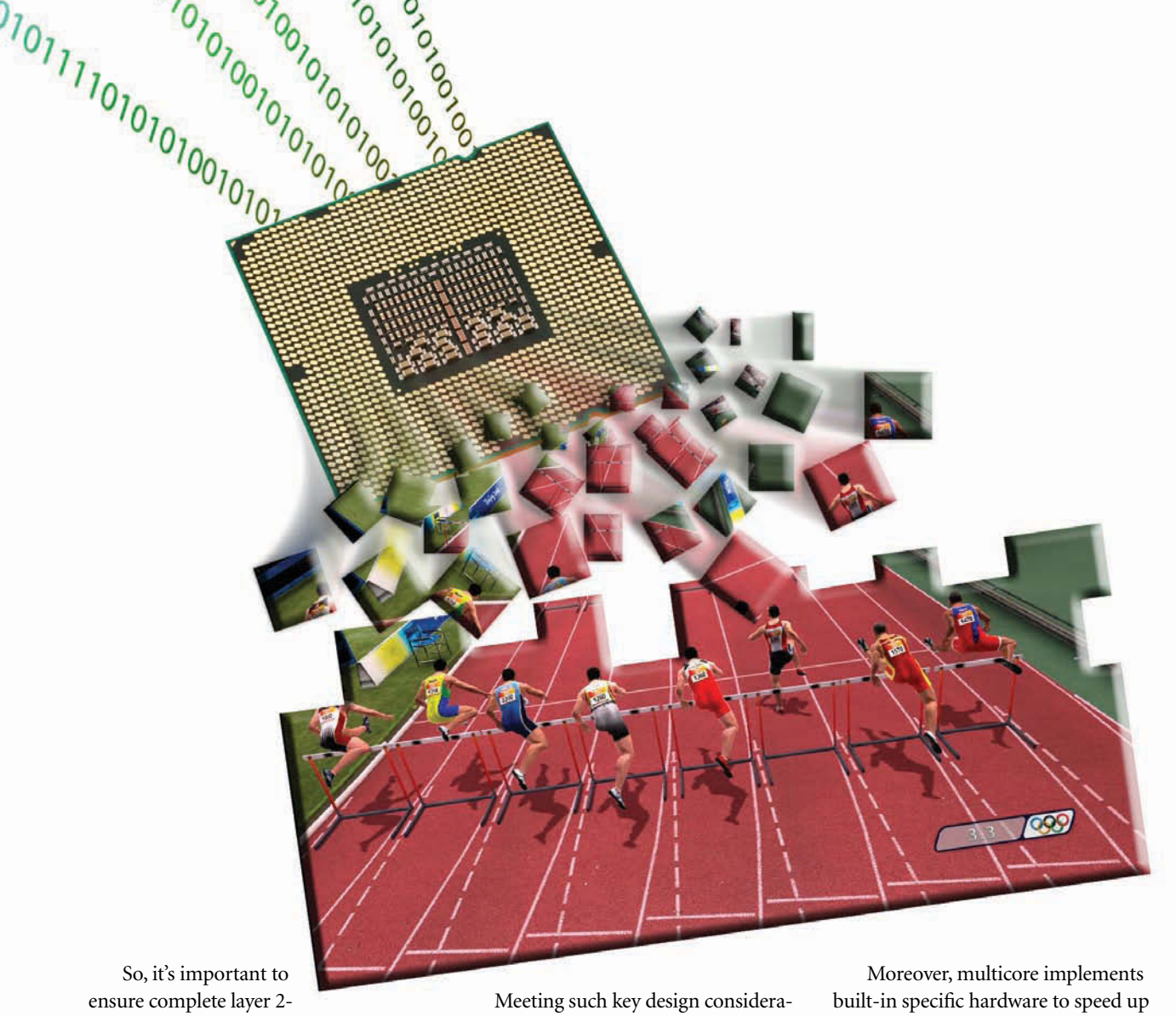
Multicores can transform IP-based wired/wireless networking

BY ERIC CARMÈS

The explosion of IP-based applications and services heralds in a convergence of telecommunications and IT toward IP architectures. This simplifies the networking architecture but leads to major challenges. A summary of challenges includes:

- The communications layer for the next generation network (NGN) will need tremendous performance enhancements and sophistication.
- IT systems (such as application servers and billing systems) will have to manage multi-10 Gbps traffic.
- The IP layer to be implemented in each piece of equipment becomes a critical component.
- Green computing will be essential to limit power consumption.

Important technical design complexities are inherent in progressing toward multicore, but the advantages are many. Developing embedded networking software for multicore can be perceived as complex because standard software cannot fully benefit from multicore improvements and requires some long and costly redesign phases for each protocol. No matter the roadblocks, the shift from running a processor at a higher frequency, as previously the only solution to increase processing capabilities, is well underway. Rightly so, as multicore allows several processors working in parallel to do the load while keeping power consumption under acceptable limits.



So, it's important to ensure complete layer 2-4 software functionalities for multicore. This includes:

- A new need for embedded networking software specifically for multicore that includes an efficient fast path architecture to make the best use of multicore performance according to the number of cores; It can be achieved using a specific multicore execution environment (MCEE), which provides APIs to implement lock-free packet processing that optimizes memory bandwidth contention and leads to unrivalled performance, compared with a standard operating system.
- A flexible distribution of the control plane/slow path/fast path over the cores, and a complete synchronization between these three elements running in separate environments.

Meeting such key design considerations will significantly reduce design challenges, time, and costs when migrating applications previously running on single-core architectures to instead run on top of multicore environments.

MULTICORE HERALDS A SOLUTION

Recent technology improvements brought upon by multicore CPUs can achieve the expected level of performance required to manage demanding network performances while keeping power consumption under control. Multicore processor technology was introduced a few years ago. It is based on an instance of a generic processing unit (core) that is duplicated to build a complete chip (multicore). These cores are interconnected through *ad hoc* communications hardware, such as message rings or switching fabrics, to be able to handle parallel processing and exchange information between cores.

Moreover, multicore implements built-in specific hardware to speed up packet processing, such as network interfaces, crypto-processors, pattern matching processors, or dedicated queues for efficient quality of service (QoS) management. A multicore architecture is by essence scalable as larger configurations can be built by interconnecting several multicore CPUs.

Multicore offerings are now maturing and provide different flavors of products to address specific market requirements:

- **Highly integrated multicore processors** (CPU, network interface, and built-in accelerators) provide a complete and integrated networking solution for packet processing. A couple of popular vendors fall into this first category with mature offerings.
- **Generic multicore processors** provide a less integrated networking solution but more processing capa-

bilities for applications, and there are vendors that fall into this category. Generic multicore processors can provide an excellent solution to integrate both application and middle-range packet processing capabilities with optional assistance from external coprocessors for encryption or pattern matching analysis.

These two solutions can be combined to provide the optimal networking method and the best migration scenario to minimize the impact on applications. For instance, highly integrated multicore processors can be used to provide a high-performance packet processing front-end while generic multicore processors are used for application processing. So, multicore processors do significantly contribute to the telecommunications/IT convergence by providing powerful hardware solutions to process IP packets efficiently.

DESIGNING NETWORKING EQUIPMENT WITH MULTICORE

The main concepts of efficient network equipment architectures were defined some years ago with the demand for designs of high-speed routers to face the explosion of Internet traffic. Now, this architecture has been extended to new services, such as L2 protocols, security, mobility, and multicast. IP-based equipment can be partitioned into three basic packet-processing elements: data plane, control plane, and management plane shown in **Figure 1**.

The *data plane* is a subsystem of a network node that receives and sends packets from an interface, processes them in some way required by the applicable protocol, and delivers, drops, or forwards them as appropriate. The *control plane* maintains information that can be used to change data used by the data plane. Maintaining this information requires handling complex signaling protocols. Implementing these protocols in the data plane would lead to poor forwarding performance. A common way to manage these protocols is to let the data plane detect incoming signaling packets and locally forward them to the control plane. Control-plane signaling protocols can update data-plane information and inject outgoing signaling packets in the data plane. This architecture works because signaling traffic is a very small part of the global traffic. The *management plane* provides an administrative interface into the overall system. It contains processes that support operational administration, management, or configuration/provisioning actions.

For equipment that has to process high bandwidth and deeply inspect the incoming packets, the data plane and control plane have to be managed by different processors. In such a case, functions implemented at the data-plane lev-

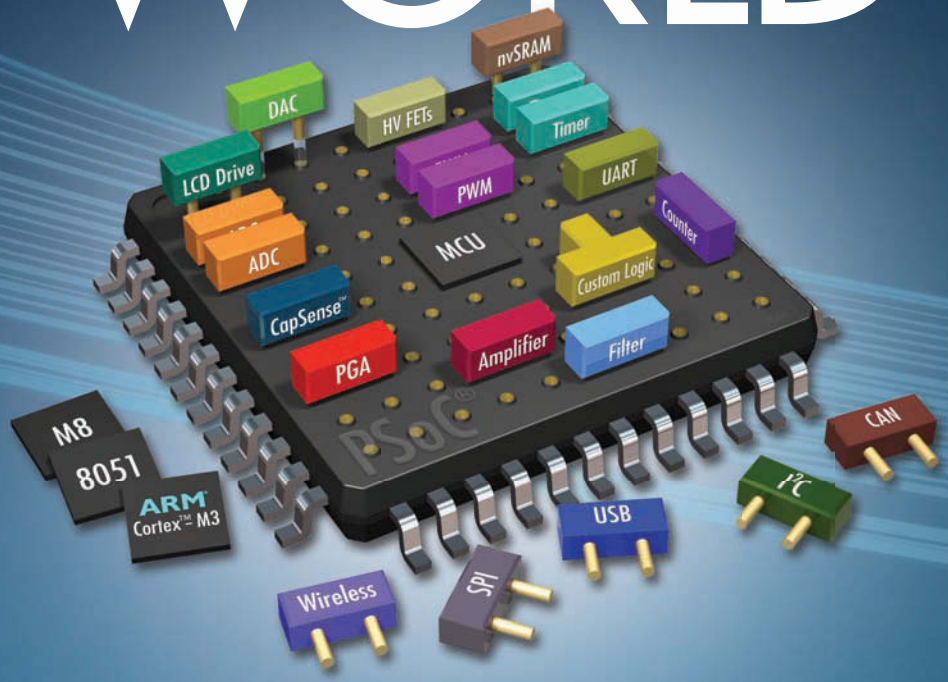
The world's only programmable system-on-chip just got even better. Cypress's PSoC® 3 and PSoC 5 architectures give you the flexibility to adapt to changing requirements with the precision analog and performance boost you need to tackle 8-, 16- or 32-bit applications. Learn more now at **www.cypress.com/go/psoc**

CYPRESS
PERFORM

WE'VE CHANGED THE WAY YOU CHANGE THE WORLD

INTRODUCING PSoC 3 & PSoC 5

- Up to 100 DMIPS and 80 MHz performance
- High-precision, 20-bit resolution, integrated analog (± 0.1% V_{REF})
- Programmable logic with up to 48 cascading datapaths
- Simplify designs, reduce BOM costs, protect your IP and get to market faster



www.cypress.com/go/psoc

Network equipment architecture.

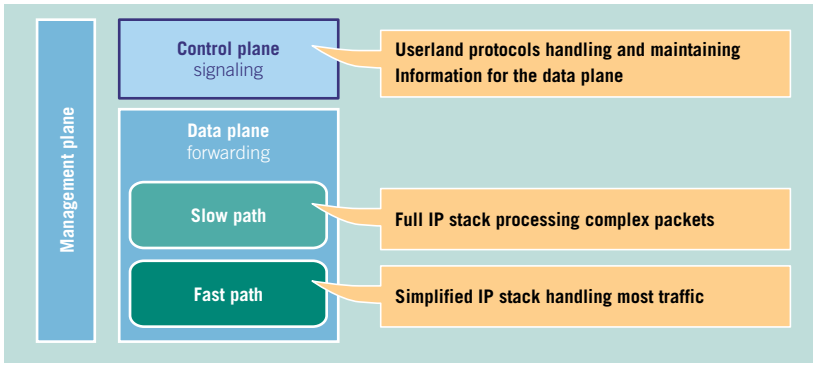


Figure 1

el are performance-oriented; algorithms are simple but have to be efficient.

The data plane, itself, can be subdivided into two parts:

- **Fast path:** The *fast path* handles all mechanisms that require per-packet processing. For instance, it determines to which port a packet should be forwarded or it encrypts/decrypts packets if IPsec security association is defined. However, some complex packets are not processed at this level and are redirected to the slow path through exceptions. (**Figure 2** shows the fast-path architecture.)
- **Slow path:** The *slow path* handles the few packets that are too complex to be processed by the fast path. Generally, the slow path implements a complete IP stack and handles most of the exceptions of the fast path. The slow path and control plane can be co-localized on the same processor.

Until now, ASIC and microcoded processors were the best solutions for data plane implementations. Control-plane processors can be more generic as they're used to process complex finite state machines to implement signaling protocols. When using multicore technology, software can share all the cores in the so-called SMP (symmetric multiprocessing) mode. For networking applications, it requires a TCP/IP stack efficiently running in the SMP mode (shown in the left side of **Figure 3**).

As shown on the right side of **Figure 3**, it is also possible to use the flexibility of multicore to distribute cores between two environments, one for the control plane and the slow path and one for the data plane. Within a single chip, some cores can be dedicated to implement a high-performance fast path. The fast path runs outside the operating system, using services of a low-level executive environment (MCEE, also called "bare metal"). This architecture optimizes fast-path performances as a standard operating system, such as Linux, brings performance bottlenecks when the number of cores grows. The programming model used in a bare-metal environment is the "Run to Completion" model: as soon as a packet has been allocated to a core, all the processing on this packet will be done by this core. A pipeline model that assumes a packet may move from one core to another core, for an elementary part of the packet processing, is not ideal for high-performance packet processing as it would induce some delays on each packet to wait for an available core. The slow path and the control plane can be implemented in a standard OS environment on the rest of the cores using the operating system in SMP mode. The distribution of cores between both environments is done at boot time.

Fast-path architecture

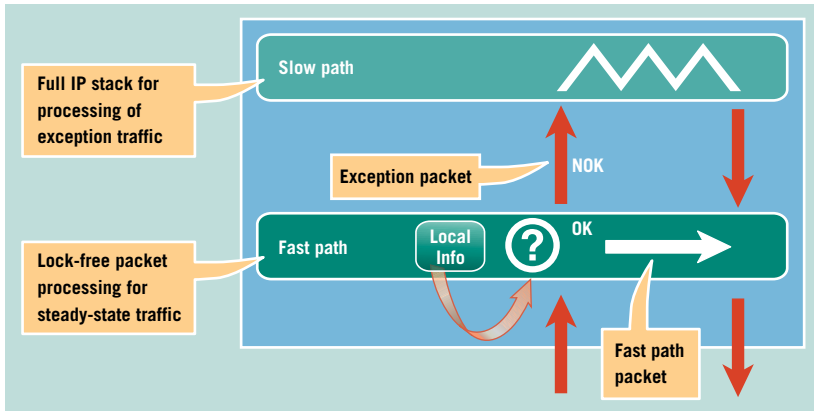


Figure 2

MULTICORE BRINGS SOFTWARE COMPLEXITY FOR PACKET PROCESSING

Processing capabilities, a high level of integration with lower power consumption, and flexibility are important advantages provided by multicore technology. They are key features to handle increases in network bandwidth and to implement complex packet processing that provides an expected level of service.

Nevertheless, to fully benefit from multicore technology improvements, there are necessary changes to be made to the embedded software architecture and this becomes more complex than with legacy single-core architectures. The software and networking complexities expand as follows:

- A multicore-based system is by essence a multiprocessing system with several processing units running in parallel.
- Having different execution environments to achieve the highest level of performance brings complexity.
- Software functions are distributed between different environments and require efficient synchronization mechanisms without performance penalties between the different software parts, to provide complete functionalities.
- Use of specific multicore-based solutions could have a major impact on applications as it will require changes in applications to benefit

Integration of fast path with the operating system.

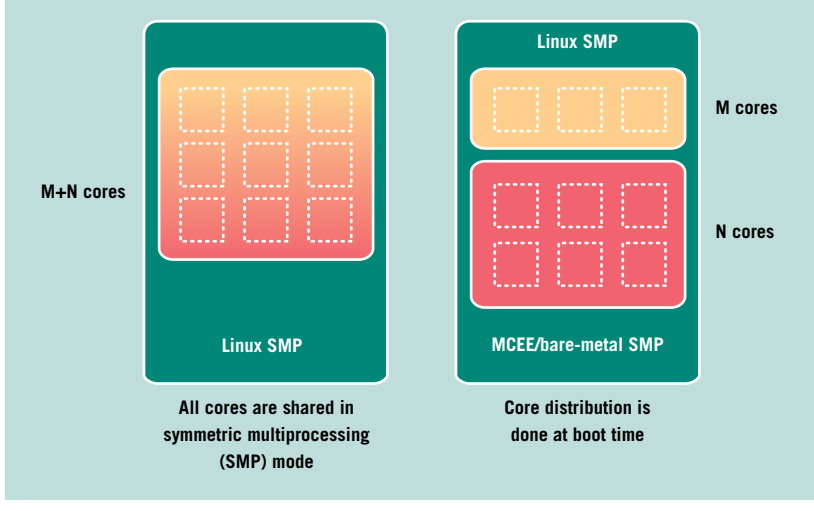


Figure 3

Integration of fast path with the operating system,

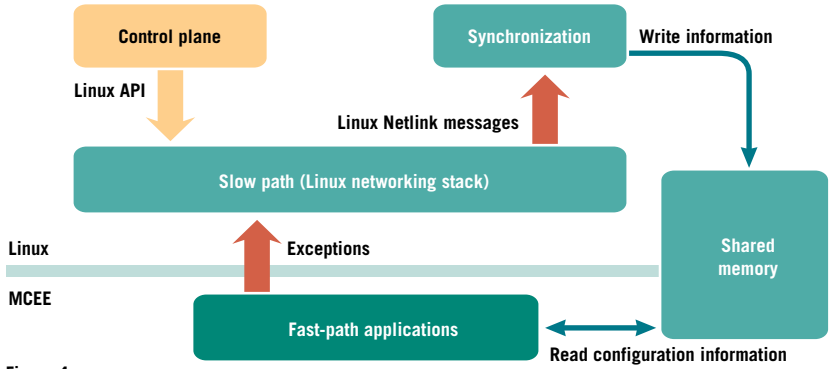


Figure 4

from multicore. The multicore packet processing architecture has to be designed to ease a migration path to multicore and software reuse.

HIGH-PERFORMANCE MULTICORE PACKET PROCESSING

Software and networking complexities create new needs for better and easier integration. Therefore, new requirements for multicore-based packet processing embedded software become necessary:

- Fast-path packet processing should be specifically designed so that it can take full advantage of the performance that is achievable with multicore processors.
- A comprehensive and easy to use set of networking features is required to remove networking integration complexities. To do so, networking feature sets must address VLAN, Link Aggregation, GRE encapsulation, GTP and IP over IP tunneling, Layer 3 forwarding with virtual routing management, routing and virtual routing, per Packet QoS and Filtering (ACLs), IPsec, NAT, IKEv1 and IKEv2 for security, multicast, and more.
- Packet processing must be fully integrated with the control plane OS to maximize software reuse (thus removing complexities in integrating applications), simplify integration, and to limit embedded software multicore design complexities.
- Embedded networking and integration features should be made to run on market-leading multicore platforms including integrated and generic multicore processors with unified high-level APIs for interfacing built-in or external accelerators such as crypto engines. In this way, platform flexibility can be made an option and seamless integration, platform to platform, is more easily achievable.
- Embedded software should provide an open architecture to ease integration of differentiating and value added features.

Open architecture.

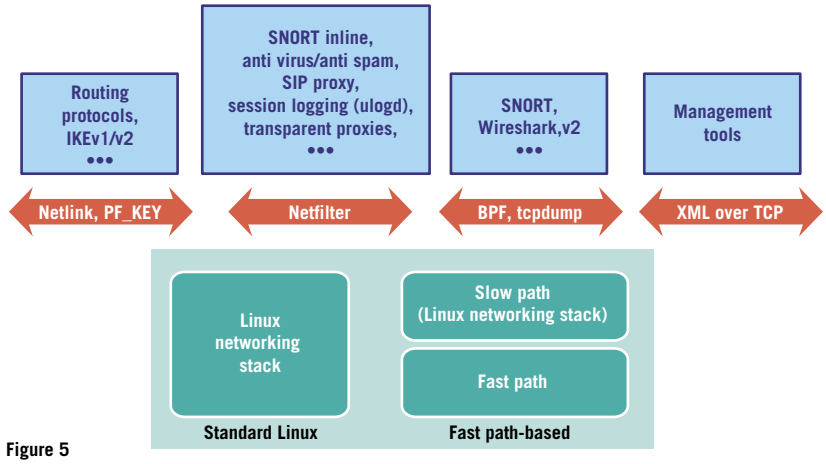


Figure 5

FAST-PATH IMPLEMENTATION

As mentioned earlier, an efficient fast-path implementation requires the redesign of each protocol using the exception strategy described in Figure 2 to provide the highest level of performance. As it has to be done on each protocol to avoid any performance penalties, this requires a significant amount of work.

An efficient fast-path implementation also means making the best use of hardware engines that can assist software processing, such as crypto engines for IPsec or hardware queues for QoS management. Although a full-IP–based architecture greatly simplifies the communication layer, there are still a large number of protocols to manage to provide L2 to L4 features. They include:

- Layer 2: VLAN, Link Aggregation, Bridging, PPP, L2TP, and so forth
- GRE encapsulation, GTP and IP over IP tunneling
- Layer 3 forwarding with management of virtual routing tables
- Routing and virtual routing
- VRRP
- Per-packet QoS and filtering (ACLs)
- NAT
- Flow inspection
- IPsec, IKE, and IKEv2 for security
- ROHC
- TCP offload
- Mobile IP
- All features for both IPv4 and IPv6

All these protocols are tightly coupled. Adding protocols has an impact on performance as it requires adding tests in the software code. So, an efficient fast path implementation needs to provide mechanisms that make it possible to use

only the required protocols to maintain maximum efficiency.

INTEGRATION WITH THE OS

A fast path-based architecture needs to split the implementation of protocols in several parts. As a consequence, all these parts need to be synchronized to have a complete implementation.

Figure 4 provides a detailed view on how the fast path running under a MCEE can be integrated with the OS and how information in the fast path, slow path, and control plane are synchronized. Fast-path applications (in other words, the fast-path part of each protocol) find required information (such as ARP tables, routing tables, and security associations) in a shared memory. As explained in Figure 2, as soon as a packet cannot be processed by the fast path, an exception is raised to inject the packet at the right place in the Linux stack. The slow path is not aware the packet is coming from a complex fast path; it is considered as a standard network interface. In cases such as an ARP resolution, the Linux stack manages it by itself. If it is a control-plane packet like a routing or an IKE packet, it is forwarded to the right control-plane protocol. The issue to be solved is to make it possible for the control-plane protocol to update the shared memory with the following constraints:

- No change in the control-plane protocol due to the multicore architecture
- No penalty performance in the fast path

For this purpose, a synchronization software module should be inserted to listen to updates coming from the control plane using Netlink messages. Then a translation can be done to messages to configure the information for the fast path in the shared memory through a fast-path driver. Such architecture avoids modifying an existing control-plane protocol. It means for instance that an existing routing protocol running on a standard Linux networking stack will run on a multicore-based architecture seamlessly without any modification. The update of the shared memory should rely on lock-free mechanisms because it's not possible to interrupt the fast path. Updates of tables have to be done with great care so that it doesn't disturb fast-path packet processing.

Possible migration scenarios to multicore.

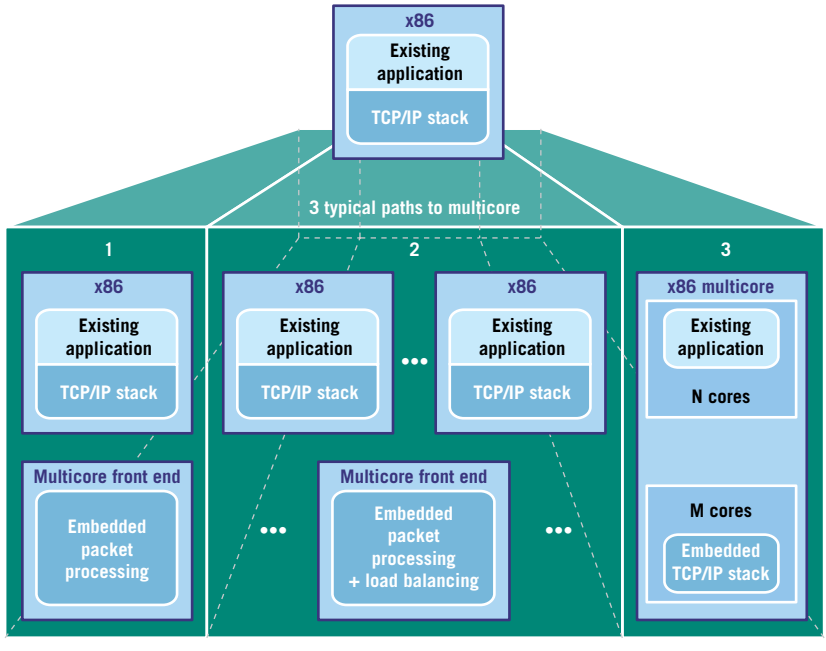


Figure 6

Retrieving exact statistics from the complete system is also part of the synchronization between fast path, slow path, and the control plane. If no specific mechanism is added to merge fast path with slow-path statistics, an SNMP agent, for instance, will only provide slow-path statistics. An additional module, therefore, has to be developed to make this merge transparent. The same issue can be highlighted for well-known debugging tools such as tcpdump that need to work transparently on a fast-path–based architecture.

Implementing such mechanisms allows reusing existing control plane and application software. They significantly reduce development work and simplify migration to multicore architectures.

PORTABILITY

As mentioned earlier, different flavors of multicore processors exist, each with different capabilities. To run on these different categories of multicore processors, packet processing software needs to be portable, as much as possible, to different architectures.

Of course, there are limitations to portability as each multicore technology may embed some specific hardware accelerators. So, standard APIs should be used whenever possible. For instance OCF (Open Cryptographic Framework) makes it possible to interface both synchronous and asynchronous crypto drivers. APIs should also be defined to interface hardware queues to provide QoS services.

OPEN ARCHITECTURE

Developing complete equipment relies on the integration of different pieces of software. Moving from a standard architecture to multicore-based architectures should not hinder integration of different pieces of software. If we use Linux as an example, there are some well known APIs to integrate routing protocols, IKE, security applications, ToIP applications, and management tools as shown in **Figure 5**.

The fast-path architecture has to implement specific features to provide standard Linux APIs to applications as well as synchronization mechanisms as described in Figure 4. Doing so, development of applications on top of a fast-path–based networking is straightforward.

A SMOOTH MIGRATION TO MULTICORE

According to equipment or applications requirements, migration to multicore can use different paths:

- Multicore can be introduced for a new design.
- The migration of an existing application to scale to meet an increase in network traffic has software reuse as a major requirement. **Figure 6** describes possible migration scenarios without a complete redesign of the application or equipment.

Designers can take three typical migration path scenarios to scale, for instance an existing x86 application. In Scenarios 1 and 2, multicore packet processing can provide standard features for front-end packet processing, such as L2, IP forwarding, IP packet reassembly, and IPsec. All of these features are normally resource-consuming on a standard processor but are not so with multicore maximizing packet processing. Multicore packet processing can also load balance the traffic (Scenario 2) to have several x86 application servers running at the same time. Using multiple packet processing front ends can extend processing capabilities and provide the application with high-availability features. Scenario 3 makes moving existing applications to x86 multicore possible by dedicating some cores to the applications and the others to networking functions.

Within the different scenarios, multicore can also provide some specific packet processing to identify sessions in the traffic (NAT sessions for large scale NATs, SIP flows for soft switches, SCTP sessions), informing applications when specific packets are received and significantly offloading packet processing from the application. Combining the first two scenarios with Scenario 3 also makes sense for high-end applications. ■

Eric Carmés is founder and CEO of 6WIND. He has more than 15 years' experience in IP standards and architectures and is an expert in current and next-generation IP technologies and protocols and multicore. Carmes holds a Master of Science degree from both INSA (French University for Applied Sciences) and ESE (French Electrical Engineering University). He can be contacted by email at *eric.carmes@6wind.com*.