

Super-scalable Network Processing

By

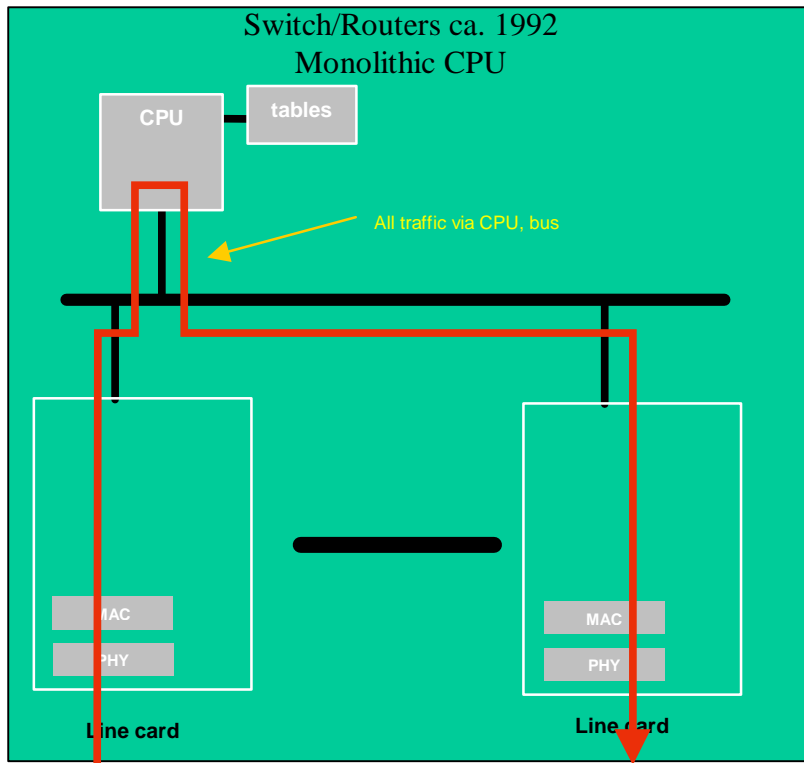
Soew Yin Lim

AMCC Switching and Network Processing

As the rampant growth of the Internet has continued to accelerate, the need for ever-greater levels of router speed and functionality is presenting system builders with both lucrative opportunities and significant challenges. In response to increased network processing requirements, router architectures have steadily evolved from early pure software implementations, through limited-function ASIC-based hardware-intensive designs, and ultimately to optimized Network Processor (NPU) based architectures that bring together both performance and flexibility. However, even with NPU-based optimization, continued demands for more speed (e.g. OC-192, OC-768) are pushing the need to use parallel processing with multiple NPU cores in order to meet wire-speed throughput requirements. In this paper, we will review the evolution of router architectures to their current levels and will explore in-depth the issues, challenges and benefits of new multi-core NPU architectures, with particular emphasis on the need to assure both scalability and manageability of multi-core designs.

Evolution of Router Architectures

In the early 1990s, routers were typically built around a central general-purpose processor with a series of line-cards that consisted of little more than physical port extensions (each containing a MAC/PHY interface). All of the traffic passing through the box was handled directly by the central CPU. The primary advantages to such architectures were that they provided the ultimate in flexibility and could be programmed in software to provide virtually any required set of functionality. However, the forward scalability of designs based around a monolithic CPU processor is inherently limited. Even the rapid advances in semiconductor performance predicted by Moore's Law have been unable to keep pace with the exponentially accelerating demands of network technologies. This has been especially true since the optical networking revolution has driven bandwidths and data flow into unprecedented new realms. With single-CPU systems limited to total throughput levels of approximately 60K packets per second (pps), they offer fairly limited usefulness in today's networks for anything beyond smaller scale LAN-oriented end-user environments.



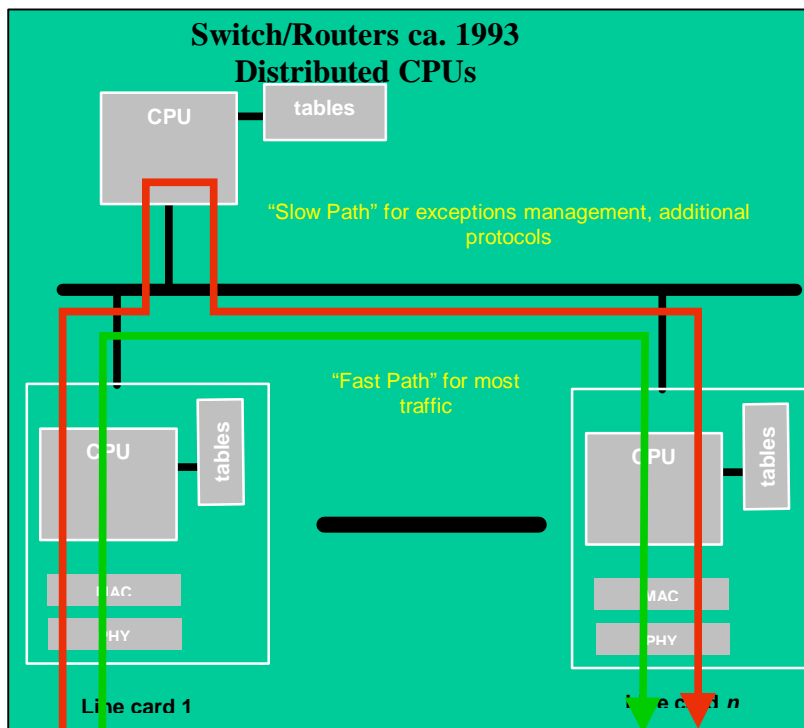
Plus

- Highly programmable (feature flexible, future proofed)

Minus

- Limited Speed/Scalability
~60k pps for entire box
4 x 10 Mbps Ethernet

The next step in router evolution focused primarily on replicating the CPU function at various points throughout the system. In addition to the central CPU, subsidiary CPUs were also added to each line card in order to distribute the decision-making load throughout the system and thus derive higher aggregate performance.



Plus

- Improved performance: "Fast Path" traffic bypasses single central CPU

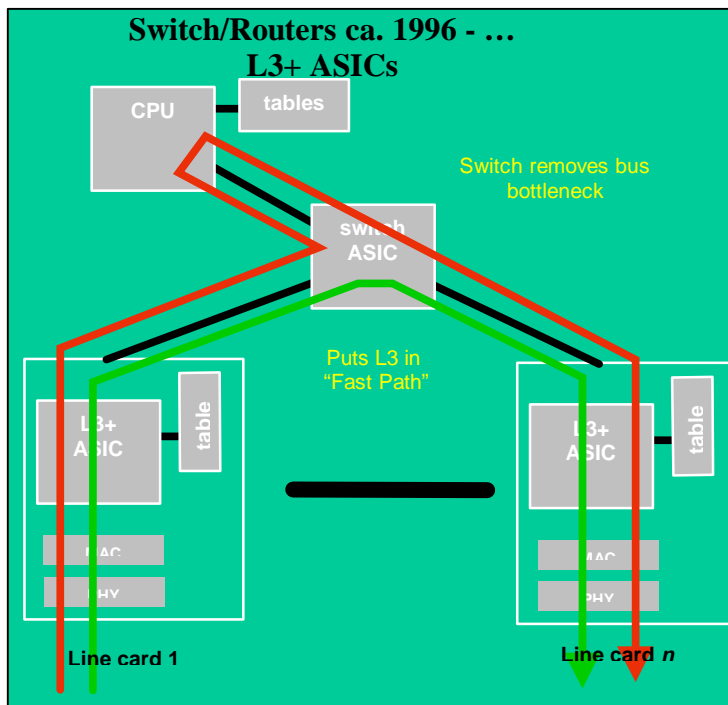
Minus

- Complexity, cost
- Still slow by today's standards
~2-300k pps for entire box
20 x Ethernet
2 x Fast Ethernet

These multi-CPU designs often took on the characteristics of “a network in a box” with each CPU essentially acting as an independent router. While this did bring overall system performance to a higher level, it did so at the cost of significantly more complexity and redundancy, such as the need to manage complete routing software and update duplicate routing tables across all of the CPUs.

However, one very key concept that evolved from these designs was the idea of separating “fast” and “slow” paths between distributed processors. This concept enabled data flow of packets to proceed unimpeded along the “fast” path, while management functions such as routing table updates are handled separately across the “slow” path. Along with the separation of fast and slow paths came the associated concept of dividing different functions between the central CPU and the line-card CPUs in order to simplify overall system management. For example, it quickly became clear that it was more efficient to centralize route-table calculations in the main CPU and only keep cache copies of IP forwarding tables with the line-card CPUs. These basic concepts of “division of functions” and “separation of data planes and control planes” have continued to form the architectural basis for subsequent system designs.

With central processors assuming responsibility for overall network topology awareness and management of routing information, the line-card processors could be further optimized for delivering faster forwarding speeds. This led to the introduction of hardwired, fixed-function ASICs to handle simplified line-card functions at much higher speeds, thus introducing the concept of wire-speed forwarding. Because of ASICs’ fixed-function limitations, initial designs were targeted for L2 switching operations rather than full L3 routing, with all exceptions having to be handled by the central CPU. Even though ASICs with some L3 capabilities were introduced toward the end of the 1990s, the inherent limitations of fixed-functionality and the longer development cycles for ASICs have made them impractical for addressing today’s dynamically-changing, multi-protocol environments, where policy-based management has become a critical factor.



Plus

- Wire-speed routing!
- Multiple protocols (e.g. IP, IPX, etc.)
- Low cost (LANs)

Minus

- TTM for ASICs
- Still fixed-function (some configurability, but not fully programmable)
- Limited queuing for QoS

ATM, multicast, VoIP, etc.), even highly optimized NPU architectures are migrating toward multi-core, parallel processing implementations. However, with port densities, board space and power dissipation constraints continuing to be significant concerns, it is vital for multi-core architectures to assure optimal instruction set efficiency and per-core processing “headroom” in order to minimize the number of cores required to meet each higher level of wire-speed performance.

For example, current multi-core architectures for handling single-stream OC-48c speeds span the range from only 2 cores to as many as 18 cores. Those instances that require more cores are generally the result of approaching the problem as one of “brute force” with general-purpose processors, rather than optimizing from the ground up. Obviously, those architectures that have been optimized from the ground up for networking and which require only 2 cores can provide both higher per-processor “headroom” and a significantly lower level of power dissipation for the proscribed amount of throughput. Even more importantly, this built-in architectural efficiency also allows for scaling to OC-192 10-gigabit levels with as few as 6 cores on a single die, thus providing even more compelling space, cost and power advantages as raw speeds continue to increase.

Processor Core Architectures: Assuring Scalability and Manageability

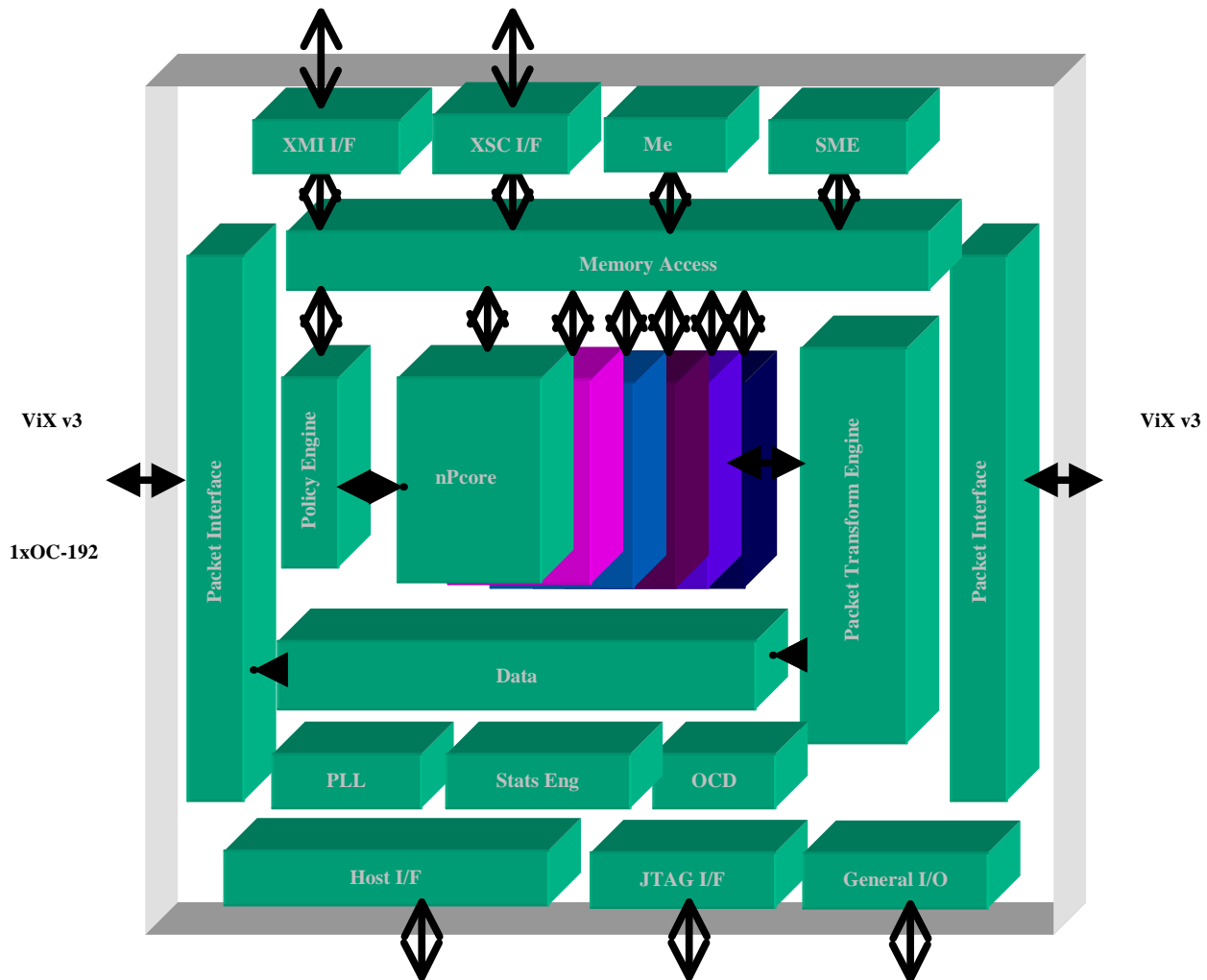
The ability to build highly optimized multi-core NPUs must start with the fundamentals, such as the instruction set efficiency, single-cycle interaction between NPUs and coprocessors, and programming efficiency.

Because today’s dynamic, multi-service, decision-based network processing challenges do not lend themselves to conventional cache-based methods, NPU-based architectures are optimized from the processing pipeline to the Instruction set to handle network-specific tasks, such as packet parsing, header searching, and packet modification, on a continuous flow, wire-speed basis. By focusing on a “NISC” (Network Instruction Set Computing) architecture, NPUs are able to eliminate many of the extraneous overhead functions, such as accumulator units, arithmetic calculations, etc., that are useful in RISC but completely unnecessary for fast network processing. In NISC, every instruction is optimized for networking, including the ability to create specific “meta-instructions” such as replacing a packet’s MAC address in a single cycle instead of the many cycles required in a general-purpose environment.

Another fundamental aspect of creating highly scalable multi-core architecture is optimization of the relationships between the NPU cores and all associated on-chip coprocessors, as well as off-chip resources. For example, the ability give multiple cores simultaneous access shared to memory structures and single cycle access to a specialized on-chip Policy Engine for packet classification can dramatically boost processing throughput and efficiency without adding undue design complexity. As shown below, the intelligent integration of multiple NPU cores with a variety of on-chip support

resources can leverage parallelism and performance without compromising either efficiency or scalability.

Multiple NPU cores Sharing Common Resources



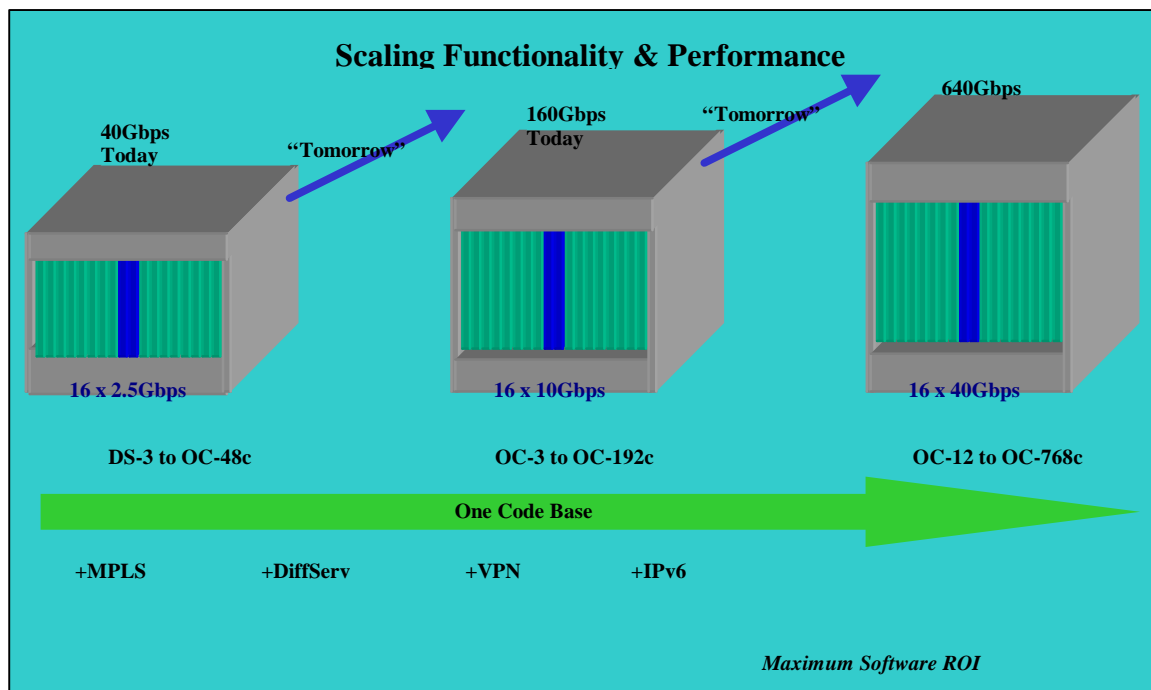
From a software perspective, the use of comprehensive programming environments that unify all processors as if they were a single unit is critical for assuring scalability and manageability as the number of cores increases to meet higher speed requirements. This typically has not been possible with standard RISC cores because their focus on providing rich general-purpose software instruction sets makes them inherently unsuitable for efficiently handling multi-processor data flows. The system designer therefore must become involved with manually partitioning the code to maintain load balancing between the processors and to maximize utilization of all the CPU cores. In effect, the system designer becomes a critical variable in optimizing the low-level

design for wire-speed performance. As the sheer number of CPU cores increases horizontally in order to address higher speeds, the burden on the system designer increases exponentially, as does the risk of performance problems in the software implementation.

In contrast, by giving system designers a single, logically-unified programming environment, NPU architectures enable them to stay out of the low-level optimization issues between multiple NPU cores and to treat the whole Network Processor function as an efficient monolithic logical device, regardless of the number of cores. Ultimately, the use of a single unified programming model is not something that can be added-on to general-purpose processors but rather must be efficiently designed into the processor architecture and instruction set from the beginning.

Looking Ahead to Future Challenges

With next-generation system requirements continuing to escalate in terms of per-port line-rate speeds, aggregate throughput rates, and multi-protocol packet processing requirements, the super-scalability of fundamentally efficient multi-core NPU designs will provide the foundation for tomorrow's successful designs. As shown below, the ability to move from OC-48c to OC-192c and OC-768c while simultaneously scaling aggregate throughput to 160Gbps – 640Gbps levels will be critical factors for remaining competitive. At the same time, the need to smoothly adapt to expanding service requirements, such as MPLS, DiffServ, VPNs, IPv6, etc., will necessitate a highly extensible coding model.



Because system designers no longer can afford the luxury of long development cycles or custom hardware-intensive ASIC-based architectures, they have rapidly embraced the advantages of maximizing the return on their investment across extensible software designs and optimized NPU architectures. By leveraging such inherently super-scalable fundamental technologies, forward-looking system developers have already laid the foundation for meeting future market requirements, while minimizing redesign challenges and maximizing on-going profitability.