Arteris™
THE NETWORK-ON-CHIP COMPANY

NoC Solution Interconnect for Managing Increasing SoC Complexity

By K. Charles Janac, President and CEO of Arteris Inc.
and Philippe di Crescenzo, Director of Engineering

As System-on-Chip (SoC) semiconductors become more complex, interconnect technology needs to keep pace. SoCs with 50 top-level IPs, or with huge subsystems that are bigger than an entire SoC of the previous generation, are forcing a rethink of SoC assembly.

The first segment that is reaching these levels of complexity is the mobility ecosystem, in which computers are being turned into cell phones, such as the *iPhone*, or cell phones are being turned into computers, such as the *Nokia N95*. During the last few years, the Network-on-Chip (NoC) concept has emerged as an attractive way to win the SoC complexity challenge.

A NoC is a multi-layer packet-based transport system, operating within a chip in a way that is loosely similar to the way an Internet operates. At the technical level, however, the comparison stops there, for the following reasons.

Feed-forward algorithms typically implemented for an Internet are not used in a NoC because of the excessive latency they produce, unlike the NoC's use of wormhole routing algorithms to minimize on-chip latency.

Similarly, deterministic routing is applied in a NoC to provide proper handling of verification and reliability issues, because Internet-suitable adaptive routing would be deficient in these areas.

It is commonly held that the asynchronous approach to NoC implementation has its place, but not in mainstream SoCs, because of the NoC's impact on proven RTL-to-Layout EDA flows, physical IP libraries, and SoC test procedures.

The Arteris Globally Asynchronous, Locally Synchronous (GALS) methodology, however, provides the benefits of the asynchronous approach, with its partitioning of complex SoCs into synchronous islands of power and voltage, without disrupting EDA and physical IP methodologies.

The GALS approach combines practical implementation of the best networking concepts with pragmatic handling of submicron SoC interconnect issues.

All major SoC components such as CPUs, DSPs, DMAs, Video engines, hardware accelerators, and so on, have in common the generation of load and store transactions.

An efficient interconnect solution therefore must support the simultaneous generation of a large number of transactions of various sizes changing transition states and undergoing packaging as they are optimized for handling.

It is this fundamental requirement that clearly distinguishes a NoC from classical terrestrial networks, such as the Internet.

SoC designers and architects must deal with the interconnection of many various IP blocks. Although this interconnect typically represents only about 6–10 percent of total SoC area, it is very challenging to design, and accounts for most of the integration and timing problems.

Common among such problems are long wires and their associated timing closure issues, late-known or unclear floorplan constraints, and multiple frequency and power domains. It is not unusual for entire projects to be delayed, or in fact, cancelled, because of interconnect related-issues.

A true NoC solution therefore must be deployed at multiple stages, from specification down to physical design of the SoC. It must provide solutions to the issues faced by a whole range of users, from SoC architects and designers, to verification and layout engineers.

The remainder of this article examines a few of these commonly-identified issues and explains how a well-designed NoC can address them.

On-chip interconnect requirements

Many well-known standards defining socket protocols, such as AXI™, AHB™, and OCP™, are being used by a wide community of designers. IPs which output these protocols are often proven on multiple designs. Many companies also use proprietary protocols and sockets across projects, or even company wide. Consequently, in order to facilitate design re-use and interoperability, there is a need to connect IPs that use different socket methods. From a NoC perspective, communication between an "AMBA™ initiator" and an "OCP target" should be supported transparently whatever the differences in semantics between the two protocols.

An interconnect in a complex SoC must also cope with the QoS requirements of all the integrated IPs, the traffic between which is often classified as follows:

Guaranteed Bandwidth or Real Time: Throughput-critical traffic, for example, to fill a display memory.

Processor: Latency-critical traffic, such as for a CPU, for which a lost cycle can never be recovered.

Best effort: Traffic which exploits remaining bandwidth.

Part of the interconnect challenge is to handle these different IP traffic needs.

Another part of that challenge is efficient transport, one of the most important aspects of the NoC because a properly defined NoC transport protocol can significantly improve global system performance. In comparison, using bus-based socket protocols for transport leads to very inefficient implementation because, at the transport layer, wires are long and routing becomes difficult. It is critical therefore to keep the number of wires for transport to a minimum, which results in much easier and less costly buffering and pipelining. Designers must keep in mind that any on-chip communication system that has not been designed for wire optimization will create problems for high-performance designs.

Certain new SoC designs are becoming more and more complex and expensive to develop. Such designs often have multiple derivatives that address multiple market requirements. These platforms, which impose new constraints on topology because of the wide spectrum of applications they support, need to evolve quickly to address market needs. The efficient NoC therefore must support any

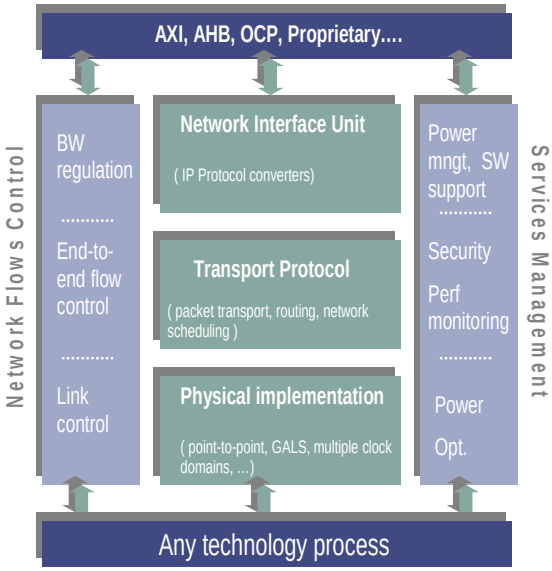
interconnection topology (2DMesh, NUMA, octagon, clustered) and be suitable for late-stage specification changes.

An effective NoC solution must also be flexible enough to address a problem discovered in layout without having to redesign the entire interconnect architecture. Moreover, for optimal control, an interconnect should not place restrictions on the granularity of clock and power domain partitioning.

Finally, as mentioned earlier, because an on-chip interconnect is dominated by wires, not gates, it should be extremely wire efficient, in order to avoid congestion.

Meeting the requirements of a pragmatic NoC

Given a basic NoC architecture, as illustrated by the following figure.



This architecture resembles a "real" IT network because of its distinct use of the following three layers:

Physical: The *medium* that carries information (32, 64, or 128 bits, GALs, Chip-to-Chip links).

Transport: Provides routing and arbitration of packets, as required by QoS.

Transaction: Converts transactions into NoC packets.

A key aspect of efficient NoC system design is the use of Network Interface Units (NIU), which employ a mechanism that maps initiator transactions into packets, and converts packets back into transactions for the target. This mechanism is more efficient than encapsulation, and provides true interoperability for multiple protocols being used on the same chip. Moreover, transaction destinations are decoded by NIUs, making the transport more efficient.

Arteris *NoC Solution* provides native support for all existing standard socket protocols (AXI, AHB, OCP, custom) and interfaces with third-party memory controller IP providers.

The following example illustrates NIU operation:

> Given an OCP Initiator running at 200 MHz and generating a "Write increment of 5 words (no response) at address 0x00010008."

In this example, the NIU:

- decodes the target (a 32-bit AHB slave), and
- maps the transaction into a NoC "Write Increment" type packet at address 0x0000008.

With the final destination specified, a Transport with the following elements can be built:

- a 64-bit data-path,
- reduced to 32 bits (size converter),
- followed by a bisynchronous FIFO,
- to reach the target NIU running at 133 MHz.

The target NIU will map the NoC packet onto a write incremental unspecified of ten cycles on the AHB socket.

Quality of Service and Memory

To obtain an effective NoC, initiator requirements for quality of service must be handled together with efficient access to memory controllers. From the memory controller perspective, memory efficiency is obtained by smart scheduling of transactions. From the initiator perspective, QoS is ensured by throughput services guaranteed by the NoC.

To handle these requirements, the Arteris NoC IP library contains QoS-dedicated units, among which:

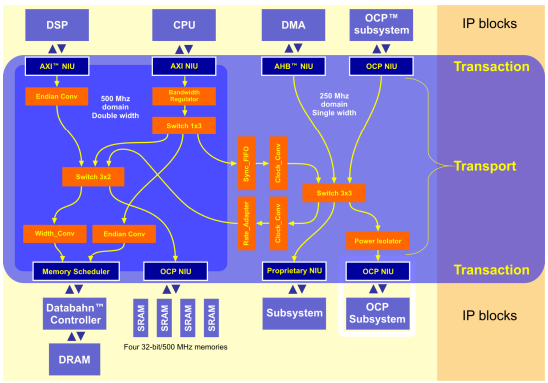
- The *Bandwidth Regulator* unit, which guarantees throughput for a given initiator.
- The *Pressure* unit, which embodies a mechanism that allows QoS-related data to be transported through the NoC from the initiator to the target and across multiple levels of switches.
- The *Memory Scheduler* unit, which optimizes DDR access at the target, based on packet priority and other configurable criteria such as bank swapping, page hits, Read–Write turns, and so forth. The unit makes it possible to obtain a tradeoff between latency and throughput, based on dynamic priorities managed by the NoC.

For example, software engineers typically ask for maximum CPU performance, but in compensation are unable to define a limit for memory bandwidth consumption. In consequence, the platform architect needs to define minimal guaranteed bandwidth for CPUs. Any excess bandwidth is made available in "best effort" mode. Real-time traffic, however, which is more regular and predictable, always gets priority when buffering capacity falls short. QoS handling, therefore, must be totally supported by the memory scheduler.

Memory interleaving is an advanced technique for memory handling that is used to deal with multiple DDR interfaces and variable burst sizes in the system. The technique makes it possible to handle small memory bursts, for example, the 16-byte bursts typical of MPEG computing or data cache refill, together with large ones, such as 128-byte bursts from DMA rasters or Ethernet Gigabit interfaces.

Obviously, 32-bit DDR interfaces handling 16-byte bursts is not a particularly efficient solution. A flexible NoC therefore must be able to address both multiple 16-bit DDR interfaces, which efficiently handle small bursts, and interleave larger bursts on different memories. This capability greatly simplifies the initiator's task, and enables easier configuration of the final product.

Efficient transport



Self-contained packets are at the root of an efficient NoC solution. NIUs are responsible for generating these packets, which carry both transport control information and payload. This approach is extremely wire-efficient, and has very little impact on latency.

This packet-based approach make it possible to completely separate request and response networks, which operate independently. This prevents deadlock, and allows for specific optimization when potentially dissymmetric traffic occurs. Because the Arteris NoC is fully stateless,

it can scale without limitation to fill any design needs.

To understand how the NoC handles a typical basic transport problem that occurs in most of today's multicore systems, consider the following example.

> Given four 32-bit traffic flows at 200 MHz, aggregated into a single 64-bit traffic flow at 400 MHz.

The NoC must provide a way to merge this traffic into a single stream without inserting unnecessary wait states. In the Arteris system, the solution lies in various NoC IP library units, such as *rate adapter*, *size converter*, and *FIFO*, that have been specifically designed for this purpose.

To deal with multiple, cross-over clock domains, or multi-die chips, designers resort to using different physical layers. As a result, the transport layer must be independent from the *medium*, or physical layer. In these circumstances, link widths and frequencies are the easiest to change. When NoC operation crosses multiple frequency domains, designers can use a mesosynchronous link, which provides on-chip transport of data plus clock on the same long wires, without skewed routing. In case of multi-die designs, the Arteris Chip-to-Chip link can be used to transport NoC packets within the context of a physical SERDES-type interface, for example, the PIPE standard from PCI Express.

These options also ensure that designers can always respond to unavoidable changes in specification and floorplan.

NoC integration is best performed at the last stage of design, when timing margins are low and routing constraints are tough. The following Arteris NoC features provide an effective way to deal with layout changes:

Point-to-Point connection: Vital to any bus-based interconnect solution, which creates layout problems because of unpredictable loads (fanout + wires). This is true not only for the high performance part of the interconnect, but also when a large number of blocks (peripherals) are connected. Bus structures are making the interconnect much more exposed to last minute changes, such as load and timing variations.

Wire efficiency: Of utmost importance for the back-end, measures the average throughput-per-wire. Requires an efficient transport protocol operating over a minimal number of wires and able to support multiple physical layers. Performing MUX–DEMUX operations in the NIU, that is, at the network peripheral, is key to easing backend operations, avoiding routing congestion, and facilitating timing convergence.

Pipelining: Last but not least, provides the capability to insert a pipeline stage anywhere in the NoC (NIU, Transport), if required. This Arteris feature allows designers to fine tune interconnect performance (frequency, latency), unlike most other competing interconnect solutions, which have predefined pipelining structures that lead to severe limitation of performance.

NoC Design Tools

NoC-compatible design, verification, and debug tools are critical for productivity and time-to-market. Even the finest NoC architecture will be of limited use to designers unless it comes with a full set of support tools. To avoid a production adoption barrier, these tools must integrate seamlessly with existing EDA tool flows. If designers are expected to fully exploit the benefits of a NoC solution, support for existing EDA standards such as *RTL*, *SystemVerilog*, *System C*, and others, is vital, not to mention NoC IPs that are

interoperable with mainstream production EDA tools.

Conclusion

In complex SoCs based on deep submicron processes, the interconnect has to be much more than just a set of raw wires. The NoC is the optimal solution to various issues such as low power efficiency, IP interoperability, architecture innovation, and challenging performance requirements, which traditional bus-based approaches are incapable of handling any more.

This brief overview has touched on topology, QoS, and layout, key areas in which the Arteris NoC makes a clear difference. The NoC can be further enhanced with security, test interface verification, and software application debug, to provide even more added value for complex SoC designs.

When choosing a NoC solution, it is vital to carefully consider design requirements and challenges in order to obtain a SoC that performs correctly, efficiently, and as close to performance requirements as possible. The Arteris NoC can help achieve this, even in the most challenging designs.

AXI, AHB and AMBA are trademarks of ARM Holdings. SystemC is a trademark of the Open SystemC Initiative. OCP is a trademark of OCP-IP. iPhone is a trademark shared by Cisco® Systems, Inc. and Apple®. Nokia and Nokia N95 are trademarks of Nokia® Corporation.