Design, Synthesis, and Test of Networks on Chips

Partha Pratim Pande
Washington State University

Giovanni De Micheli
Ecole Polytechnique Fédérale de Lausanne

Cristian Grecu, André Ivanov, and Resve Saleh
University of British Columbia

Editor's note:
For networks on chips to succeed as the next generation of on-chip interconnect, researchers must solve the major problems involved in designing, implementing, verifying, and testing them. This article surveys the latest NoC architectures, methods, and tools and shows what must happen to make NoCs part of a viable future.

—Grant Martin, Tensilica

SoC design in the coming billion-transistor era implies the extensive use and seamless integration of numerous semiconductor IP blocks in the form of processors, embedded memories, and smart interconnects. Such systems will behave like multiprocessors and will require a corresponding design methodology for both their hardware and software implementations. Power and cross-chip-signaling constraints are driving development of new design methodologies to incorporate explicit parallelism and provide a more structured communication fabric. Many researchers[1,2] have rightfully argued that arrays of interconnected processors that form the basis of new multiprocessor SoC platforms (the so-called MP-SoC platforms) will dominate future designs. Furthermore, to meet the communication requirements of large SoCs, a network-on-a-chip (NoC) paradigm is emerging as a new design methodology. Therefore, system design must encompass both networking and distributed computation paradigms and provide underlying communication infrastructures that allow effective integration of functional and storage blocks.

Many current SoC designs contain numerous processors for applications such as set-top boxes, wireless base stations, high-definition TV, mobile handsets, and image processing.[2] Recent literature discusses new trends in the design of communication architectures in multicore SoCs.[1,3] In particular, researchers suggest building multicore SoCs around different regular interconnect structures originating from parallel-computing architectures.[3] Custom-built application-specific interconnect architectures are another promising solution.[4] Various trade-offs regarding latency, throughput, reliability, energy dissipation, and silicon area requirements characterize such communication-centric interconnect fabrics. (See the "Terminology" sidebar.) An application's nature will dictate the selection of a specific template for the communication medium.

Figure 1 shows a representative set of interconnect templates proposed by different research groups. Kumar proposed a mesh-based interconnect architecture called Cliché (Figure 1a).[5] Grecu et al.[6] describe an interconnect architecture based on the butterfly fat-tree (BFT) topology for a networked SoC; they also describe the associated design of the required switches and addressing mechanisms (Figure 1b). Karim et al.[7] proposed the Octagon multiprocessor SoC architecture (Figure 1c). Octagon is a special case of a more general class of networks called Spidergon.[8] Benini and Bertozzi[4] describe an irregular application-specific NoC interconnect template (Figure 1d).

A salient feature of NoC architectures is the decoupling of the communication fabric from the processing and storage elements.[1] This lets designers optimize the communication medium independently of the functionality, using different levels of abstraction. By viewing a complex SoC as a micronetwork of multiple blocks, designers can borrow models and techniques from networking and parallel processing and apply them to SoC design methodology. (See the "Programming models for NoCs" sidebar on p. 406.) The micronetwork must ensure energy efficiency and quality-of-service (QoS) requirements such as relia-

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

Terminology

Regardless of their specific implementation, NoC architectures require evaluation in terms of their throughput, latency, energy dissipation profiles, silicon area overhead, and wiring complexity. Throughput is the maximum load the network can physically handle, and it determines the system's aggregate bandwidth. We define transport latency as the time (in clock cycles) that elapses between a message injection into the network at the source node and the end of packet reception at the destination node. In Figure A, throughput and latency characteristics for a typical NoC appear as a function of the injection load. When the injection load approaches the throughput saturation limit, latency starts to increase exponentially.

When data travels on the interconnection network, both the interswitch wires and the logic gates in the switches toggle. This results in energy dissipation and adds to a SoC's overall energy budget. When evaluating the feasibility of these interconnect schemes, designers must consider the area overhead required for switch blocks and network interfaces. Interswitch wires are another source of silicon area overhead. Depending on their lengths, they might have to be either pipelined or buffered through repeater insertion to keep the interswitch delay within one clock cycle.[1] Consequently, designers should consider this additional buffer and register area.

The problem of estimating wire area complexity involves determining the longest wire segments that might arise in each architecture and their distribution. Long wire segments block wiring channels and force other wires to become longer. In a NoC environment, the interswitch wire segments are the longest on-chip wires, except for clock, power, and ground wires. The structured nature of NoC-based interconnects lets designers predict the interswitch wire lengths with reasonable accuracy. Although quantifying the overhead attributable to wiring complexity might be difficult, analyzing the distribution of interswitch wire lengths can provide a first-order estimation.

Reference

1. P.P. Pande et al., "Performance Evaluation and Design Trade-offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Computers*, vol. 54, no. 8, Aug. 2005, pp. 1025-1040.

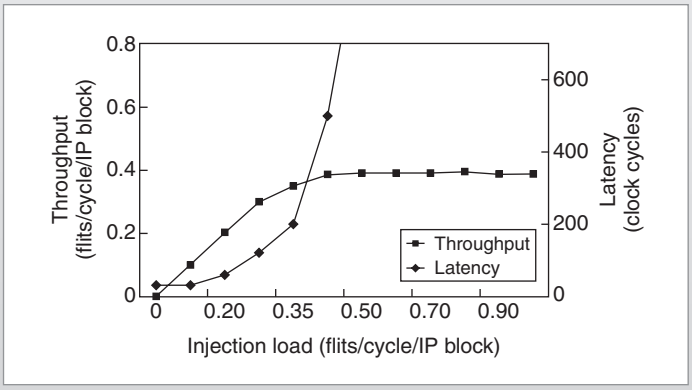

Figure A. NoC throughput and latency characteristics. We measure throughput in number of flits/cycle/IP block and latency in clock cycles.

bility and guaranteed bandwidth and latency under the limitation of intrinsically unreliable signal transmission media. Such limitations are due to the increased likelihood of timing and data errors resulting from crosstalk, variability of process parameters, and environmental factors such as electromagnetic interference and soft errors.[1]

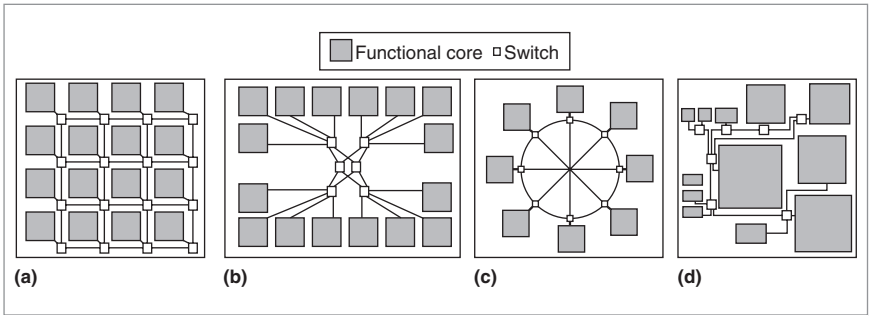

Figure 1. Network-on-a-chip (NoC) interconnect architectures: Cliché (a), butterfly fat-tree (BFT) topology (b), Octagon MP-SoC (c), irregular application-specific template (d).

To become viable, the NoC paradigm requires support by CAD tools through the creation of specialized libraries, application mapping tools, and synthesis flows.[9] Commercial design frameworks (such as Sonics[10]) and proprietary design frameworks (such as STBus[2] and Æthereal[11]) are scarce for NoCs. A few research design tools address specific problems. Still, the novelty of NoCs makes it hard to establish a taxonomy or

Programming models for NoCs

The range of functional IP blocks extends from application-specific processors, to general-purpose RISCs, to I/O and memory blocks, and all must comply with a common programming platform. A programming model for parallel systems like NoCs is a description of the basic components, their properties and available operations, and their synchronization.[1] The two primary parallel computing models are the parallel random-access machine and the message-passing model. The ST Microelectronics MultiFlex multiprocessor SoC programming environment focuses on both these models. It includes a distributed-system object component message-passing model and a symmetrical multiprocessing model using shared memory. Developed specifically for multiprocessor SoCs, the MultiFlex environment maps these models onto the StepNP platform.[1] Van der Wolf et al. describe a task transaction-level (TTL) interface, which application developers can use for developing parallel application models by integrating hardware and software tasks on a platform.[2] Because it is an abstract interface, it permits the use of the TTL interface easily without knowledge of low-level implementation details, and it allows implementation of a broad range of multiprocessor platforms.

References

1. P. Magarshack and P.G. Paulin, "System-on-Chip beyond the Nanometer Wall," *Proc. 40th Design Automation Conf.* (DAC 03), ACM Press, 2003, pp. 419-424.
2. P. van der Wolf et al., "Design and Programming of Embedded Multiprocessors: An Interface-Centric Approach," *Proc. Int'l Conf. Hardware/Software Codesign and System Synthesis* (CODES + ISSS 04), ACM Press, 2004, pp. 206-217.

attempt a fair comparison. Therefore, we report on some existing research projects as case studies.

To validate functionality and performance at various abstraction levels, ranging from electrical to transaction levels, designers can port current simulation methods and tools to networked SoCs. NoC libraries, including switches, routers, links, and interfaces, give designers flexible components to complement processor and storage cores. Nevertheless, the usefulness of such libraries will depend heavily on the maturity of the corresponding synthesis and optimization tools and flows. In other words, micronetwork synthesis will enable NoC and SoC design similar to the way logic synthesis made efficient semicustom design possible in the 1980s.

Wide adoption of any new design methodology depends on its having a complement of efficient test mechanisms. Developing test infrastructures and techniques to support the NoC design paradigm is challenging. Specifically, novel DFT schemes and the design of specialized test access mechanisms (TAMs) for distributing test vectors are very important.[12] Moreover, in a communication-centric design environment like that of NoCs, fault tolerance and reliability of the data transmission medium are significant requirements in safety-critical VLSI applications.

Practical implementation and adoption of the NoC design paradigm faces multiple unresolved issues related to design methodology and technology, test strategies, dedicated CAD tools, and analysis of architectures. This article discusses these challenges, some proposed solutions, and future directions worth pursuing.

Design considerations

Although the design process for NoC-based systems borrows some aspects from the parallel computing domain, it is driven by a significantly different set of constraints. From the performance perspective, high throughput and low latency are desirable characteristics of MP-SoC platforms. However, from a VLSI design perspective, the interconnect architecture's energy dissipation profile is critical because it can represent a significant portion of the overall energy budget. Silicon area overhead resulting from the interconnect fabric is important too. The common characteristics of these kinds of architectures are that the processor and storage cores communicate with one another through high-performance links and intelligent switches, and communication design can be represented at a high abstraction level.

Switch block design

There is a definite trend toward packet-based on-chip communication. Various schemes for packetized communication are viable and must comply with local and global QoS requirements. Switching and routing schemes require switch blocks with various characteristics.[11]

Recent packet-switching trends show that wormhole switching is the solution of choice for NoCs.[3] This scheme divides packets into fixed-length flow control units (flits), with I/O buffers storing only a few flits. Thus, unlike most other schemes, this design style minimizes the buffer space in the switches, and the switches used in a wormhole technique can be small and compact. A packet's first, or header, flit contains routing information. Header flit decoding enables switches to establish the path, and subsequent flits simply follow this path in a pipelined fashion. As a result, each incoming data flit of a message

packet is simply forwarded along the same output channel as the preceding data flit, and packets needn't be reordered at their destinations. If a flit faces a busy channel, subsequent flits must wait at their current locations.

One drawback of this simple wormhole switching method is that distinct messages cannot be interleaved or multiplexed over a physical channel. Messages must cross a channel in their entirety before another message can use it. This decreases channel utilization if a flit from a given packet is blocked in a buffer. However, introducing virtual channels in the I/O ports can improve channel utilization considerably. If a packet's flit is blocked in one virtual channel, then flits of alternate packets can use the other virtual channel buffers and, ultimately, the physical channel.

Switch design also depends on the routing scheme adopted. The two broad categories of routing are deterministic and adaptive. Deterministic routing algorithms always provide the same path between a given source and destination pair. Adaptive routing algorithms use information about routing traffic or channel status to avoid the congested or faulty part of the network. In a deterministic routing scheme, switches can be fast and compact. Figure 2 shows a schematic representation of a switch consisting mainly of input/output FIFO buffers and a routing block.

High throughput requires multiple (two in this example) virtual channels. Fast, streamlined switches can also support simple adaptive routing schemes. For example, in the Nostrum NoC, the switches realize a congestion-driven deflective routing scheme for a mesh or torus network architecture. Ye et al. combined this routing scheme with wormhole switching.[13] The switches are combinational blocks, and the decision to deflect a packet that cannot be routed efficiently toward its destination is based on the analysis of traffic congestion at the neighboring nodes.

Performance evaluation and design trade-offs

NoC-based interconnect performance correlates strongly with the topology selected for implementation. These topologies fall broadly into two categories: regular architectures (Figures 1a, 1b, and 1c) and irregular, application-specific (custom) NoC structures (Figure 1d). In regular architectures, the performance level is homogeneous across the whole system. In irregular architectures, the service requirements vary widely for the different processors and storage blocks. In the case of custom-built NoC architectures, switch blocks might not be identical; their design and placement depend on the specific communication requirements. Regular network architectures are well suited for the realization of multiprocessor communication schemes. Irregular network architectures might be necessary for realizing application-specific SoCs, such as those in mobile-phone systems, where different heterogeneous blocks with varying communication requirements must be linked.

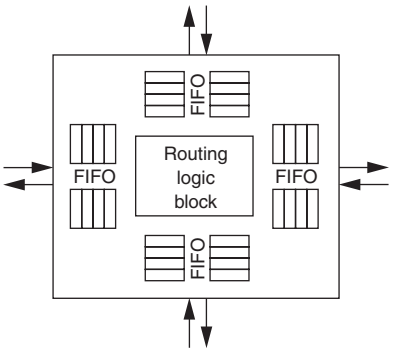

Figure 2. Switch architecture for deterministic and adaptive routing schemes.

Communication pipelining

The exchange of data among the processors and storage cores is becoming an increasingly difficult task because of growing system size and nonscalable global wire delay. To cope with these issues, designers must divide the end-to-end communication medium into multiple pipelined stages, with the delay in each stage comparable to the clock-cycle budget. In NoC architectures, the interswitch wire segments, along with the switch blocks, constitute a highly pipelined communication medium characterized by link pipelining, deeply pipelined switches, and latency-insensitive component design.[4] Link pipelining is inherently built into regular NoC topologies. Custom-built architectures require special measures to achieve link pipelining. The switches generally consist of multiple pipeline stages. The number of intraswitch pipeline stages can vary with the design style and the features incorporated within the switch blocks. However, through careful circuit-level design and analysis, designers can make each intraswitch stage's delay less than the target clock period in a particular technology node.[6]

Traffic pattern and network analysis

Evaluating the performance of NoC-based interconnects through system-level simulation requires using specific traffic patterns characterizing the data flow through the system. Designers traditionally use a Poisson-distributed injection rate when characterizing the performance of multiprocessor platforms. However, self-similar distribution is a better match for real-world SoC scenarios. Designers can model self-similar traffic by aggregating many on/off message sources. The Pareto distribution $F(x) = 1 - x^{-\alpha}$, with $1 < \alpha < 2$, fits well with this situation. For example, researchers have

observed self-similar traffic in the bursty traffic between on-chip modules in typical MPEG-2 video and networking applications.

Most researchers accept the assumption of uniform spatial distribution of traffic patterns for evaluating parallel systems. However, this is not very realistic in a SoC environment because different functions will be mapped to different parts of the SoC, and the traffic will exhibit highly localized patterns. Consider an illustration of spatial localization in a mesh-based NoC (Figure 1a): One possibility is to constrain local traffic within the four destinations placed at the shortest Manhattan distance, while the rest of the traffic is uniformly distributed among the other destinations. For example, a localization factor of 0.3 signifies that 30% of the traffic generated by a core is local. As a result of this traffic localization, a system's throughput—and hence its aggregate bandwidth—can be enhanced considerably, making it possible to transfer more data without saturating the network. Increasing the amount of traffic localization causes more messages to be injected without increasing the average energy dissipation. This happens because, on average, messages traverse fewer hops when there is greater localization. Consequently, designers should perform functional mapping to exploit the advantages of spatial locality, placing the blocks that communicate more frequently closer together. This reduces the use of long global paths and the accompanying energy dissipation.

Designers can analyze NoC performance at various levels of abstraction. Whereas general-purpose network simulators such as ns2 (http://www.isi.edu/nsnam/ns/) can achieve network simulation, abstractions that can incorporate the network as well as models for the processing and storage cores can provide more useful information. The modeling language SystemC can serve to effectively model switches and network interfaces at the transaction or cycle-accurate level. It's possible to combine such models with instruction-level, bus-level, or detailed models of the processing cores. One advantage of using SystemC is the possibility of linking software programs that can implement one or more layers of the communication protocol. The On-Chip Communication Network (OCCN) project[8] proposes an efficient, open-source research and development framework for the specification, modeling, and simulation of on-chip communication architectures. OCCN defines a universal API and an object-oriented C++ library built atop SystemC. Network emulation by FPGAs is another way to validate specific switch and network-interface implementations. Because emulation can execute two to three orders of magnitude more quickly than cycle-accurate simulation, designers can experiment with different library components and expose the components to workloads of significant length.

NoC synthesis

Synthesizing NoCs is a way to realize gate- and circuit-level models, starting from an architectural template and design constraints. Because of their novelty, NoCs have no specialized languages or formalisms for their high-level modeling. Nevertheless, structural formalisms can help designers model network topologies, and procedural languages such as SystemC and C++ can capture hardware and software behavior.

Synthesis is useful in both homogeneous and heterogeneous network architectures, but it is critical in the latter because designers must choose from, and experiment with, different topologies and parameters when searching for the best match for an application. The premise of network synthesis is that designers can realize the network by means of components such as switches, links, and network interfaces. Such components are tailored to the network and instantiated, resulting in a model that can be simulated and synthesized. There are several reasons for using a synthesis methodology for NoCs:

- Sometimes the best network architecture, protocols, and parameters for a given system application aren't known. To find the best solution, a designer must experiment with different models having various performance, energy consumption, and layout complexity trade-offs. Eventually, it will be possible to choose the network topology, protocols, and parameters automatically or with CAD tool support. In the interim, fast, automatic generation of models that the network can simulate can help designers make informed choices.
- There are many parameters to optimize in an on-chip network implementation. CAD tools can help optimize the implemented circuitry by, for example, sizing switches and links to provide adequate QoS with minimal area overhead and energy dissipation. (See the "Quality of services" sidebar.) Fine-tuning NoCs is hard and time-consuming, especially in the case of heterogeneous fabrics and networks.
- A synthesis flow allows fast design and lets designers concentrate on system issues while leaving details to the tools. When coping with the challenges of communication-centric SoCs, designers must use network synthesis to close the productivity gap in

Quality of services

Quality of services encompasses a collection of design requirements that must be fulfilled to achieve a certain performance level. Because the interconnect infrastructure provides data communication services to the constituent IP blocks, the infrastructure's design must let it maintain predictable performance under various operating conditions. In NoCs, two types of services are essential: data integrity (meaning the data is delivered without corruption) and throughput and latency services (characterized by time-related bounds). Designers achieve these services through

- contention-free routing schemes,
- error control coding,
- deadlock avoidance mechanisms, and
- appropriate flow-control strategies.

Guaranteed services require resource reservation for worst-case scenarios. As a result, resources often remain underused. Best-effort services do not reserve resources and hence provide no service guarantee. Guaranteed services should be used for critical traffic, and best-effort services for noncritical traffic. By integrating both guaranteed and best-effort services in the same interconnect, it's possible to design predictable, low-cost interconnect infrastructures.[1] In the Æthereal NoC,[1] the network interface offers a guaranteed service by providing a lower bound on throughput and an upper bound on latency, because these are the most critical components for supporting real-time communication. Æthereal implements throughput and latency guarantees by configuring connections as pipelined time-division-multiplexed circuits over the network.

Reference

1. A. Rădulescu et al., "An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, Jan. 2005, pp. 4-17.

much the same way that logic synthesis has expedited semicustom design.

Examples of NoC libraries include xPipes[4] and xPipesLite;[14] the latter is a simpler, faster, and synthesizable version of the former. The xPipes compiler (shown in Figure 3) is a network synthesis tool for xPipes, and Sunmap[9] (shown in Figure 4) is an automatic topology selection tool. Designers have used the libraries and tools to realize experimental gate-level models of complex system applications.

Both xPipes and xPipesLite rely on three major component types:

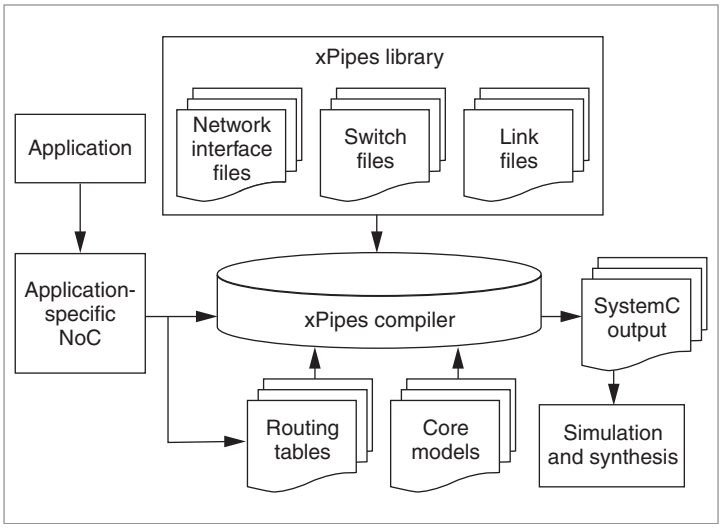

Figure 3. NoC synthesis flow.

- Network interfaces that act as wrappers for generic processor cores that must comply with the open core protocol (OCP) (http://www.ocpip.org). Acting as a protocol converter, these interfaces convert the core's I/O signals from OCP to an internal wormhole static-routed protocol. (See the "Network interfacing" sidebar on p. 411.)
- Switches that are parameterizable in terms of inputs, outputs, virtual channels, and error control schemes.
- Latency-insensitive links, which use wire pipelining to satisfy frequency constraints, also use optional timing-error control features. Bertozzi et al. provide details.[9]

Testing NoC-based systems

The test strategy for NoC-based systems addresses three problems: testing the functional and storage blocks and their corresponding network interfaces, testing the interconnect infrastructure itself, and testing the integrated system.

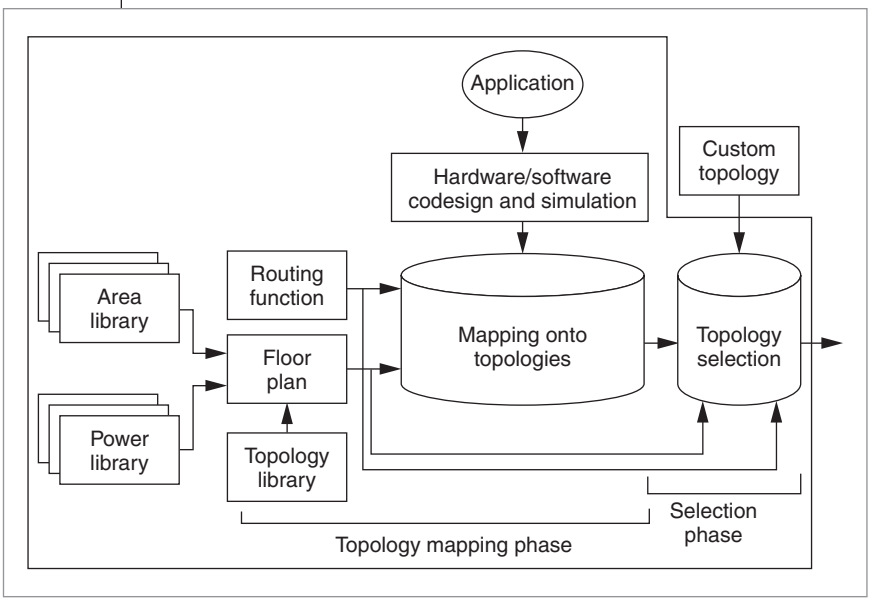

Figure 4. Sunmap design flow.

ing logic consists of a few hundred logic gates, and engineers use traditional testing methods such as scan or BIST. However, testing the FIFO buffers poses a unique challenge because many relatively small buffers are distributed all over the chip. BIST, a traditionally accepted methodology for testing FIFO buffers, is not suitable in the NoC scenario. The classical BIST approach of one dedicated BIST per FIFO block would result in an unacceptably large silicon area overhead. Consequently, a distributed BIST methodology, like that depicted in Figure 5, is more appropriate.

A distributed BIST scheme shares the read/write mechanisms, the control circuitry, and the test data source among the multiple FIFO blocks, whereas each FIFO has a local response analyzer. Researchers must investigate realistic fault models for these FIFO buffers.

Functional and storage block testing

Testing the functional and storage blocks and their corresponding network interfaces requires a test access mechanism (TAM) to transport the test data. This TAM provides on-chip transport of test stimuli from a test pattern source to the core under test. It also transmits test responses from the core under test to the test pattern sink. Cota et al. propose reusing the on-chip network as a TAM for the functional and storage cores.[12] The principal advantage of using NoCs as TAMs is the availability of several parallel paths for transmitting test data to each core and the fact that no extra TAM hardware is needed. The result is a reduction in system test time through extensive test parallelization; that is, more functional blocks can be tested in parallel because more test paths are available. One side effect of test parallelization is excessive power dissipation. Hence, both test time and power dissipation require consideration when exploiting parallelization for testing the functional blocks.[12]

Interconnect infrastructure testing

Testing the interconnect infrastructure involves two aspects: testing the switch blocks and testing the interswitch wire segments.

Testing switch blocks. The switch blocks consist of the FIFO buffers and the routing logic. FIFO buffers occupy more silicon area than routing logic, so switch block testing breaks down into two problems: testing the FIFO buffers and testing the routing circuitry. Generally, routing logic consists of a few hundred logic gates, and engineers use traditional testing methods such as scan or BIST. However, testing the FIFO buffers poses a unique challenge because many relatively small buffers are distributed all over the chip. BIST, a traditionally accepted methodology for testing FIFO buffers, is not suitable in the NoC scenario. The classical BIST approach of one dedicated BIST per FIFO block would result in an unacceptably large silicon area overhead. Consequently, a distributed BIST methodology, like that depicted in Figure 5, is more appropriate.

A distributed BIST scheme shares the read/write mechanisms, the control circuitry, and the test data source among the multiple FIFO blocks, whereas each FIFO has a local response analyzer. Researchers must investigate realistic fault models for these FIFO buffers.

Testing interswitch wire segments. This aspect involves the adoption of adequate fault models that account for deep-submicron effects. In the digital domain, device defects used to be modeled with extremely simplified models such as the stuck-at fault model. In deep-submicron technologies, crosstalk and inductive effects introduce more-complex behaviors that require more-advanced fault models for interconnect testing. Test engineers can apply the maximal aggressor fault (MAF) model proposed by Cuviello et al. to test the interswitch wire segments in NoC architectures.[15] For a link consisting of N wires, this MAF model assumes the worst-case situation with one victim line and $N - 1$ aggressors. MAF tests must execute at operational speed, which can require expensive external testers. To achieve high-quality at-speed testing of interconnects, researchers have proposed different self-test methods that use embedded BIST structures to generate MAF tests. However, these methods introduce area and delay overhead. Testing of interswitch wire segments in NoCs remains an open problem requiring further investigation.

Integrated system testing

Testing the functional and storage blocks and the interconnect infrastructure separately isn't enough to ensure adequate test quality. Interaction between the functional and storage cores and the communication fabric must also undergo extensive functional testing,

Network interfacing

The NoC design paradigm's success relies heavily on the standardization of the interfaces between IP cores and the interconnection fabric. Using a standard interface should not affect the methodologies for IP core development. In fact, IP cores wrapped with a standard interface will be far more reusable and will greatly simplify the task of system integration.[1] The open core protocol (OCP; http://www.ocpip.org) is a plug-and-play interface standard gaining wide industrial and academic acceptance. Similar to the OCP, the AMBA AXI (http://www.arm.com) is another protocol targeting the design of high-performance systems. As Figure B shows, for a core having both master and slave interfaces, a second interface packetizes the OCP- or AXI-compliant signals of the functional IP blocks. The network interface has two functions:

- injecting or absorbing the flits leaving or arriving at the functional and storage blocks, and
- packetizing and depacketizing the signals to and from the OCP- or AXI-compatible cores in the form of messages or flits.

All OCP signals are unidirectional and synchronous, simplifying core implementation, integration, and timing analysis. The OCP defines a point-to-point interface between two communicating entities, such as the IP core and the communication medium. One entity acts as the master of the OCP instance, and the other as the slave. The OCP unifies all intercore communications, including data flow, sideband control, and test-specific signals.

The burst-based AXI protocol provides a single interface definition between a master and the interconnect, a slave and the interconnect, and a master and a slave. Every transaction has address and control information on the address channel describing the nature of the data to be transferred. The data travels between master and slave using a write data channel to the slave or a read data channel to the master.

Reference

1. L. Benini and D. Bertozzi, "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, Apr.-June, 2004, pp. 18-31.

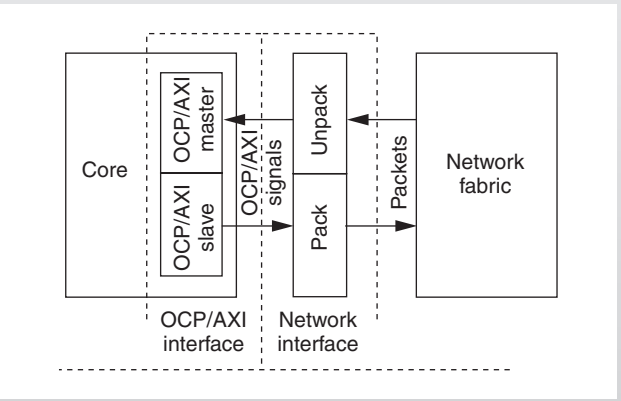

Figure B. Interfacing IP cores with the network fabric.

which should encompass testing the I/O functions of each processing element and the data routing functions.

Reliable SoC/NoC design

SoCs often reside within embedded systems, where reliability is an important figure of merit. At the same time, in deep-submicron technologies beyond the 65-nm node, failures of transistors and wires are probably caused by a variety of effects, such as soft (cosmic) errors, crosstalk, process variations, electromigration, and material aging.[1] We can generally distinguish between transient and permanent failures. Design of reliable SoCs must encompass techniques that address both types of malfunction. We address transient malfunctions first; then we analyze permanent malfunctions.

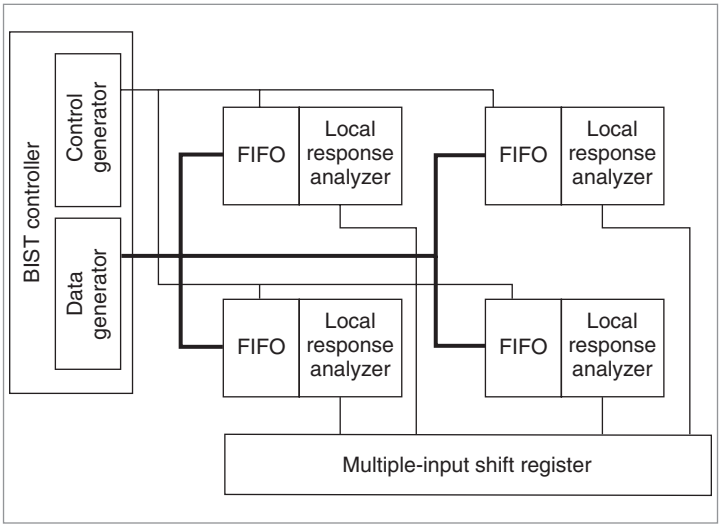

Figure 5. Distributed BIST structure for FIFO testing.

Error control coding

From a reliability viewpoint, one advantage of packetized communication is the possibility of incorporating error-control information into the transmitted data stream. Applying effective error detection and correction methods from the communications engineering domain can help in coping with transient malfunctions in on-chip data transmission. Such methods require evaluation and optimization in terms of area, delay, and power consumption trade-offs. Researchers have studied and proposed different error detecting and correcting codes for bus-based on-chip communication fabrics. They have shown that for a bus-based system, from the energy efficiency perspective, error detecting codes with retransmission are more effective than error correction.

In NoC architectures, the error recovery mechanism can be distributed over multiple hops or concentrated at the end nodes. In distributed schemes, each switch has error detection or correction circuitry such that transmission of corrupted data can be stopped or corrected at the intermediate switches. In centralized mechanisms, the retransmission of corrupted data can cause a severe latency penalty, especially when the source and destination nodes are far apart. Therefore, the trade-off related to the localization of error detection and correction involves several figures of merit, such as latency, area, and power consumption.

Fault-tolerant architectures

Permanent failures may be due to material aging (oxide), electromigration, or mechanical or thermal stress. Failures can incapacitate a processing or storage core or a communication link. Researchers have proposed various fault-tolerant multiprocessor architectures and routing algorithms in the parallel processing domain. Designers can adapt some of these solutions to the NoC domain, but they should evaluate their effectiveness in terms of throughput, delay, energy dissipation, and silicon area overhead metrics. For example, redundant standby components can serve as spare parts. On-chip networks ease the seamless integration of such components, as well as the online transition from a malfunctioning unit to a spare part. Specific on-chip network topologies can provide the SoC with multiple paths from source to destination, and this redundancy might suffice to obviate a malfunctioning link, possibly at the expense of performance.

In the realm of NoCs, designers must weigh all types of redundancies against additional layout complexity (larger chips) and increased energy consumption. Therefore, it's important to view reliable system design in conjunction with power management. Indeed, because of frequency, temperature levels, and thermal cycles, power management policies affect failure rates. Similarly, designers can use power management techniques to switch spare units on or off, thereby limiting their energy consumption impact. This area is a subject of ongoing research.

COMMERCIAL DESIGNS are integrating from 10 to 100 embedded functional and storage blocks in a single SoC, and the number is likely to increase significantly in the near future. Because of this enormous degree of integration, several industrial and academic research groups are striving to develop efficient communication architectures, in some cases specifically optimized for certain applications. The research community tends to view NoCs as an enabling solution for this level of integration. Major issues still under debate include the detailed design trade-offs and the performance optimizations accompanying this new on-chip interconnect paradigm. ■

References

1. L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, vol. 35, no. 1, Jan. 2002, pp. 7078.
2. P. Magarshack and P.G. Paulin, "System-on-Chip beyond the Nanometer Wall," *Proc. 40th Design Automation Conf.* (DAC 03), ACM Press, 2003, pp. 419-424.
3. P.P. Pande et al., "Performance Evaluation and Design Trade-offs for Network-on-Chip Interconnect Architectures," *IEEE Trans. Computers*, vol. 54, no. 8, Aug. 2005, pp. 1025-1040.
4. L. Benini and D. Bertozzi, "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, Apr.-June, 2004, pp. 18-31.
5. S. Kumar, "On Packet Switched Networks for On-Chip Communications," *Networks on Chip*, A. Jantsch and H. Tenhunen, eds., Kluwer, 2003, pp. 85-106.
6. C. Grecu et al., "Timing Analysis of Network on Chip Architectures for MP-SoC Platforms," *Microelectronics J.*, vol. 36, no. 9, pp. 833-845.
7. F. Karim et al., "An Interconnect Architecture for Networking Systems on Chips," *IEEE Micro*, vol. 22, no. 5, Sept.-Oct. 2002, pp. 36-45.
8. M. Coppola et al., "OCCN: A Network-on-Chip Modeling and Simulation Framework," *Proc. Design, Automation and Test in Europe* (DATE 04), IEEE CS Press, 2004, pp. 174-179.

9. D. Bertozzi et al., "NoC Synthesis Flow for Customized Domain-Specific Multiprocessor System on Chip," *IEEE Trans. Parallel and Distributed Systems*, vol. 16, no. 2, Feb. 2005, pp. 113-129.
10. W.D. Weber et al., "A Quality-of-Service Mechanism for Interconnection Networks in System-on-Chips," *Proc. Design, Automation and Test in Europe* (DATE 05), IEEE CS Press, 2005, pp. 1232-1237.
11. A. Rădulescu et al., "An Efficient On-Chip NI Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, Jan. 2005, pp. 4-17.
12. E. Cota et al., "Power-Aware NoC Reuse on the Testing of Core-Based Systems," *Proc. Int'l Test Conf.* (ITC 03), vol. 1, IEEE CS Press, 2003, pp. 612-621.
13. T. Ye, L. Benini, and G. De Micheli, "Packetization and Routing Analysis of On-Chip Multiprocessor Networks," *J. System Integration*, vol. 50, Feb. 2004, pp. 81-104.
14. S. Stergiou et al., "xPipesLite: A Synthesis-Oriented Design Library for Networks on Chips," *Proc. Design, Automation and Test in Europe* (DATE 05), IEEE CS Press, 2005, pp. 1188-1193.
15. M. Cuviello et al., "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," *Proc. Int'l Conf. CAD* (ICCAD 99), IEEE CS Press, 1999, pp. 297-303.

Partha Pratim Pande is an assistant professor in the School of Electrical Engineering and Computer Science at Washington State University. He was a graduate student at the University of British Columbia when he did this work. His research interests focus on design and test of NoCs. Pande has a BS in electronics and communication engineering from Calcutta University, an MS in computer science from the National University of Singapore, and a PhD in electrical and computer engineering from the University of British Columbia. He is a student member of the IEEE.

Cristian Grecu is a PhD student in the Department of Electrical and Computer Engineering at the University of British Columbia. His research interests focus on design and test of large SoCs, with emphasis on their data communication infrastructures. Grecu has a BS and an MS from the Technical University of Iasi, Romania, and an MS from the University of British Columbia, all in electrical engineering.

The biography of **André Ivanov** appears on p. 403 of this issue.

Resve Saleh is a professor in the Department of Electrical and Computer Engineering at the University of British Columbia, where he holds the NSERC/PMC-Sierra Chair. His research interests include SoC design, verification, and test. Saleh has a BS in electrical engineering from Carleton University, Ottawa, and an MS and PhD in electrical engineering from the University of California, Berkeley. He is a senior member of the IEEE.

The biography of **Giovanni De Micheli** appears on p. 403 of this issue.

■ Direct questions and comments about this article to Partha Pratim Pande, School of EECS, Washington State University, Pullman, WA 99164-2752; pande@eecs.wsu.edu.