Bringing Communication Networks on a Chip: Test and Verification Implications

Bart Vermeulen, John Dielissen, and Kees Goossens, Philips Research Laboratories
Calin Ciordas, Eindhoven University of Technology

ABSTRACT

In this article we present test and verification challenges for system chips that utilize on-chip networks. These SOCs and networks on a chip are introduced, where the NOC is exemplified by Philips' ÆTHEREAL NOC architecture. We discuss existing test and verification methods for SOCs and NOCs, and show the particular advantages of using an NOC for both testing and verifying the network, and testing and verifying the other components of the SOC. This article is concluded with our experiences with NOCs and a description of ongoing work within Philips in this emerging field.

INTRODUCTION

High-performance networking requires dedicated hardware with tremendous computational and communication performance. Network components such as network interfaces and routers are complex systems that are built in a modular fashion by combining many application-specific integrated circuits (ASICs). With increasing packet throughput, ASIC performance must increase. Moreover, trends toward differentiated services and higher quality of service require additional performance. Examples are more discerning packet classification, traffic shaping, network management, and debug. To address these issues networking ASICs must become more versatile and programmable, often evolving toward *network processors*.

To indicate that network processors and ASICs are complex systems in themselves, they are usually named *systems on a chip* (SOCs). The number of components in an SOC is growing rapidly, and the communication infrastructure on a single SOC is a major concern. In fact, on-chip interconnect will increasingly be implemented as a *network on a chip* (NOC), complete with network interfaces, routers, and packet or circuit switching. Although the distances over which communication takes place differ by many orders of magnitude, the fields of on-chip networking and computer networking are clearly related.

We show why and how NOCs are used to implement SOC communication needs, and illustrate why their implementation is different from that of computer networks. We outline Philips' ÆTHEREAL NOC, which is one such solution.

We consider how an SOC, constructed from many hardware blocks (called *cores*), and an NOC can be tested for manufacturing defects. We describe the functional verification of an SOC, verification of an NOC, and how an NOC can aid in verifying an SOC. We present how an application, mapped onto an SOC with an NOC, can be verified. We end this article with conclusions and highlight future work in the development of the Philips ÆTHEREAL NOC.

NETWORKS ON CHIP

To manage the complexity of designing an SOC containing multiple cores, design teams are adopting core reuse methodologies. These methodologies allow cores, once designed, to be reused in multiple SOCs. As a result, the complexity of building a complete SOC shifts from the design of the individual cores to the design of the communication architecture connecting the cores. To prevent the design of this communication architecture from becoming the bottleneck in the design of future SOCs, this communication architecture itself must be compositional and scalable.

A single broadcast medium, such as Amba and silicon backplane [1] buses, already can no longer deliver the required global bandwidth and latency for current SOCs. Switches such as the multilayer Amba and Prophid provide some relief, but are ultimately not scalable. Mirroring computer networks, a trend can be observed toward using networks of routers with circuit or packet switching to implement on-chip communication [2, 3]. Of particular inter-

0163-6804/03/$17.00 © 2003 IEEE

est are SOCs for networking applications that themselves use NOCs. Examples of SOCs with an on-chip mesh of packet-switched routers to implement a single-chip switch are given in [4, 5]. Karim *et al.* [6] show how a network processor for OC-768 uses a hybrid circuit-/packet-switched NOC.

Although computer networks and on-chip networks share many requirements, there are also a number of differences, which can lead to different trade-offs [2,7] and hence different architectures. Examples are:

- Quality of service beyond best effort traffic is probably more important in SOCs for consumer electronics than for Internet services, due to their embedded, real-time, often safety-critical nature [7]. Consumer applications have to be robust and require predictable performance.
- The conditions on a chip are currently more stable than off-chip. On-chip routers are considered either faulty or correct; hence, the network topology is static, and not upgradable after the chip leaves the factory.
- Routers and network interfaces of an NOC are more resource constrained than those in computer networks because they are intended for mainstream consumer products. As a consequence, the chip area must be minimized, leading to few and shallow buffers, fast and simple arbitration, limited traffic shaping, and so on.
- On-chip communication links are relatively short compared to those in computer networks. Pipelining or transmission-line effects are therefore absent. This advantage partially offsets the severe resource constraints: buffers can be small due to the tight synchronization between routers, and buffer overflow can be prevented by using flow control.
- In contrast to computer networks, inter-router wires are relatively abundant in NOCs [2]. Links can be wide, and their utilization is probably less important.

Within Philips, the need to manage present-day and future on-chip communication demands spurred the development of the ÆTHEREAL NOC. Details of its architecture are presented in the next section.

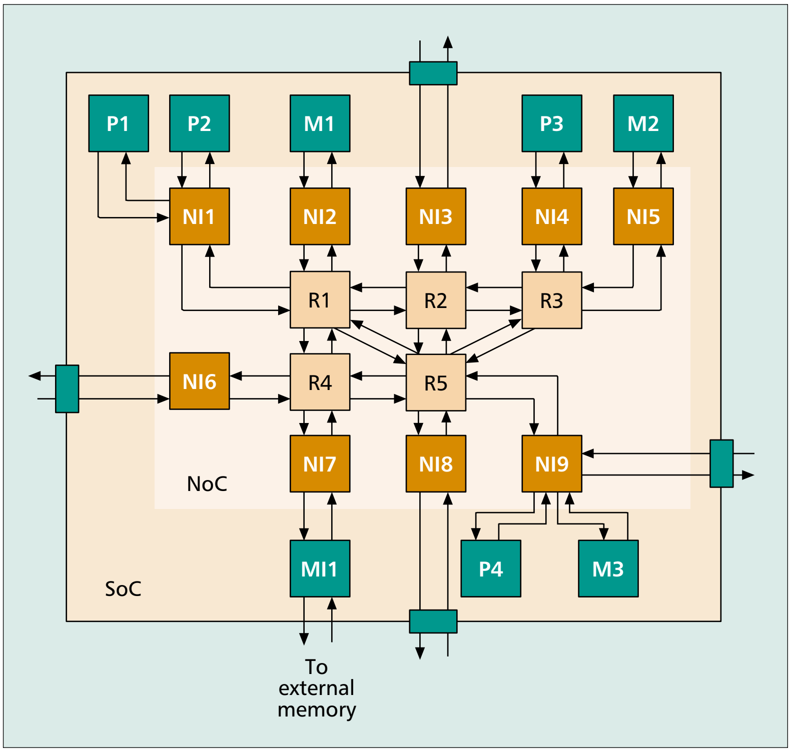


Figure 1. *Example SOC with an ÆTHEREAL NOC.*

The Æthereal Network on Chip

The Æthereal Architecture

The Philips ÆTHEREAL NOC [7, 8] addresses the communication needs of consumer electronics SOCs with real-time requirements, such as those used in digital video set-top boxes. Figure 1 shows an example SOC consisting of cores and an ÆTHEREAL NOC.

Cores communicate with each other using the NOC, and include memories (M*i*), programmable or dedicated processors (P*i*), and external memory interfaces (MI*i*). The NOC consists of routers (R*i*) and network interfaces (NI*i*), which are linked by nonpipelined wires. These routers and NIs are described in more detail in the following subsections.

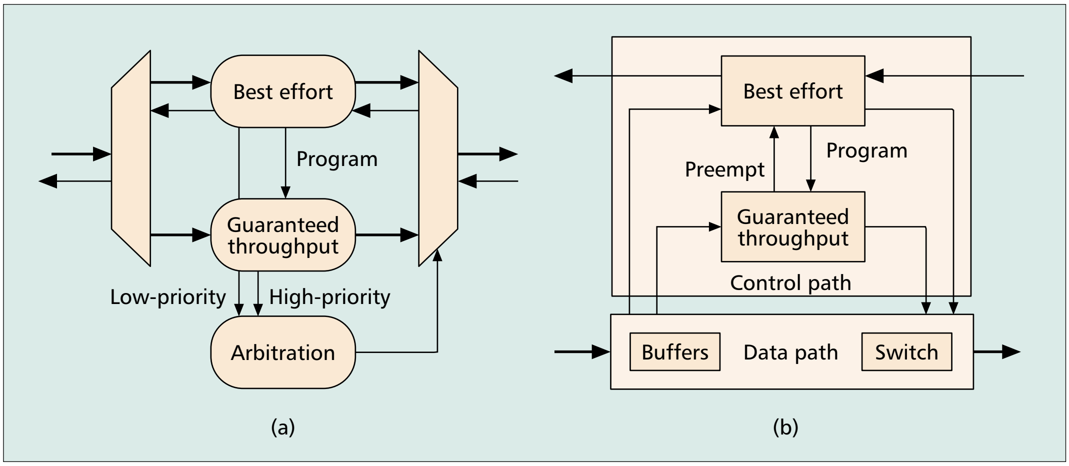
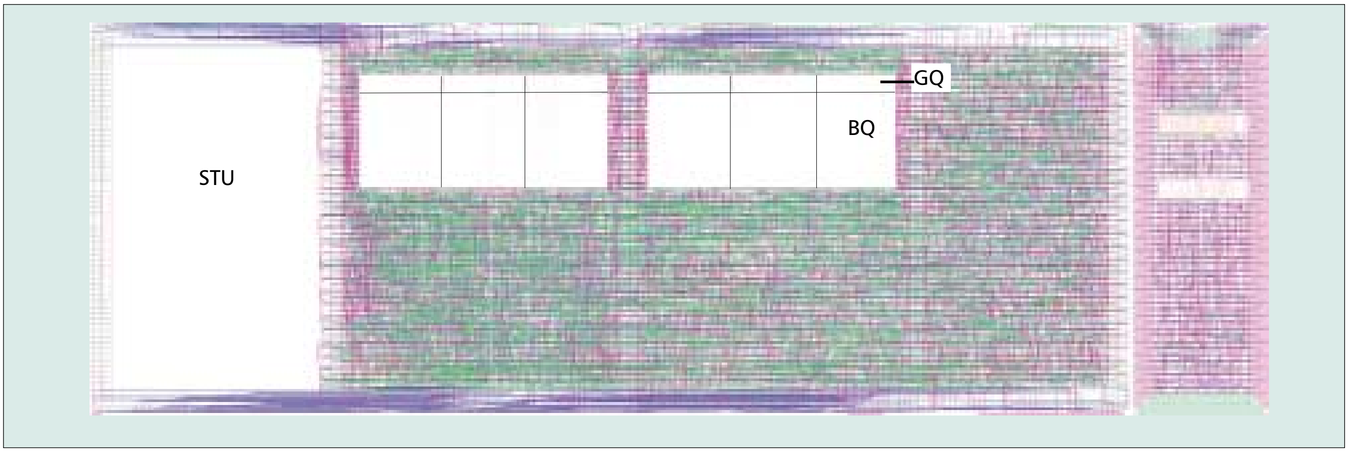


Figure 2. *Two views of the combined GT-BE router: a) conceptual view; b) hardware view.*



■ **Figure 3.** *Layout of a combined GT-BE router.*

THE ÆTHEREAL ROUTER

Conceptually, the ÆTHEREAL router module consists of two independent routers (Fig. 2a).

The *best effort* (BE) router offers uncorrupted lossless (flow-controlled) ordered data transport. The *guaranteed throughput* (GT) router adds hard throughput and latency guarantees over a finite time interval. The GT router fulfils our real-time service requirement, and combining it with a BE router ensures efficient resource utilization.

The Guaranteed Throughput Router — To offer not just statistical but also hard guaranteed latency, the network and hence the routers must be lossless. Besides this easy-to-solve property, contention must also be eliminated or quantified to be able to provide a guarantee. Rate-based and deadline-based scheduling offer guarantees [9], but require deep output or priority queues. These queues are too costly for on-chip use. A low-cost alternative is to avoid contention completely by scheduling packets at the network edge such that they are never in the same place, never there at the same time, or a combination. The ÆTHEREAL GT router uses a combination of both with time-division multiplexed pipelined circuits. Every router and network interface block contains a slot table $T(s, o) = i$, defining for a given slot s from which input i output o takes its data, if available. For this approach to work, all NOC blocks must share a common notion of time to ensure that their slot tables remain aligned. This is feasible in NOCs using mesochronous clocking (synchronous clocks with constant skew), or asynchronous hand shaking, as described by Öberg in [7, Ch. 8]. BE packets are used to program the slot tables to set up and tear down GT connections, akin to asynchronous transfer mode (ATM). This is shown in Fig. 2a by the arrow labeled *program*. Router programming packets follow the same route as the connection they program. Slot allocations can be computed and programmed at runtime in a distributed manner, or (pre-)computed offline and then configured at runtime.

The Best Effort Router — The BE router is a classical input-queued wormhole router, and uses round-robin arbitration for fairness. Data packets are never reordered in the router, and because deterministic routing is used, ordering is preserved end to end. Programming packets are shunted to a programming module in the router, and spliced in the data stream after they have programmed the slot table.

The Combined Router — As shown in Fig. 2b, the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit (including link-level flow control for the BE router) of Fig. 2a has been merged with the BE router itself. The data path, mainly consisting of the switch matrix, is shared. In computer network router architectures, the buffers of BE and GT traffic would be stored in a shared RAM. For the small amount of buffering in on-chip routers (in our case, 3 words/GT queue and 24 words/BE queue) using either RAMs or register file memories would be very area inefficient. By using dedicated GT and BE hardware first in first out buffers (FIFOs) (GQ and BQ in Fig. 3), the area of the router is reduced by two-thirds.

We synthesized an arity 5 router with a BE queue depth of 24 words of 32 bits, and a 256-slot table (STU) in 0.12 μ technology. The layout is shown in Fig. 3. It has an aggregate bandwidth of 5 × 500 MHz × 32 bits = 80 Gb/s. The area of the router is 0.26 mm^2 in a 0.12 μ CMOS process using 6 metal layers.

THE ÆTHEREAL NETWORK INTERFACE

The network interface is the bridge between a core and a router, where in general more cores can be connected to one network interface. It implements end-to-end flow control, admission control, and traffic shaping, connection setup and teardown, and transaction reordering. Like the router it contains a slot table, but has dedicated hardware FIFOs per connection.

MANUFACTURING TEST

Like all other SOCs, an NOC-based SOC has to be tested for manufacturing defects. The NOC can be considered just another core of an SOC, but it is also special in two ways:

- It is often composed of many identical subcores (routers and network interfaces).

- It occupies a privileged central position in the SOC by virtue of its interconnecting role.

In this section we explore the most relevant options for efficiently and effectively testing an SOC with an NOC.

SOC Manufacturing Test

With the design of a scalable and modular SOC comes the issue of testing for manufacturing defects. Over the last years, several advances have been made in SOC testing. IEEE P1500 [10] is standardizing a core-based test approach in which cores are wrapped in a test wrapper to allow easy testing of that core in a SOC. A so-called *test access mechanism* (TAM) is used in conjunction with test wrappers around the core to transport test data to and from a core under test. Combined they allow application-independent test access to all on-chip cores. An example of this core-based test approach is shown in Fig. 4.

In test mode, the SOC cores are distributed over four TAMs connected to the four I/O interfaces of the chip. In Fig. 4, these four TAMs are indicated by deep orange, deep green, light orange, and light green. During scan test all TAMs are used in parallel to minimize the total test time. A disadvantage of this method is that it requires additional wires to be added to the design to form the TAMs. In network chips, adding these TAMs on top of an already large number of interconnects might cause wire congestion problems during layout of the design.

Built-in self-test (BIST) is another popular approach to test mainly regular logic blocks, such as on-chip memories. Advances have been made to extend the use of BIST to other cores as well. Techniques such as random test pattern generation in combination with test point insertion or bit flipping [11] are used to test these blocks efficiently.

Typically the traditional stuck-at fault test patterns are used, complemented by both delay fault test patterns and possibly IDDQ or ΔIDDQ test patterns to meet test quality requirements.

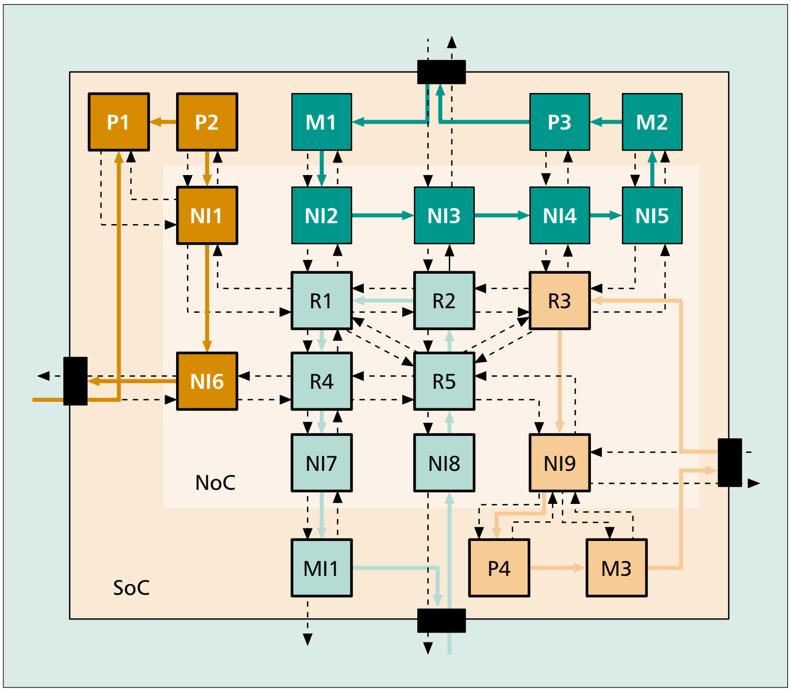


Figure 4. *Default core-based testing.*

Testing the NOC

Given the NOC's regular and hierarchical structure, it makes sense to adopt a core-based test approach. If we know the function of the core-under test (i.e., an NOC), we can utilize this knowledge to modify the standard core-based test approach to obtain a better suited test. Figure 5a shows a possible core-based approach for an SOC with an NOC.

The blocks that make up the NOC are tested first. If the NOC contains a fault, the entire SOC can be sent off for diagnosis without testing it further in the production line. This leads to a reduction in test time and consequently a savings in test cost.

The concept of test reuse can be taken one step further. While testing the NOC, all identical blocks (e.g., all routers) can reuse the same test data. This test data set is broadcast and applied

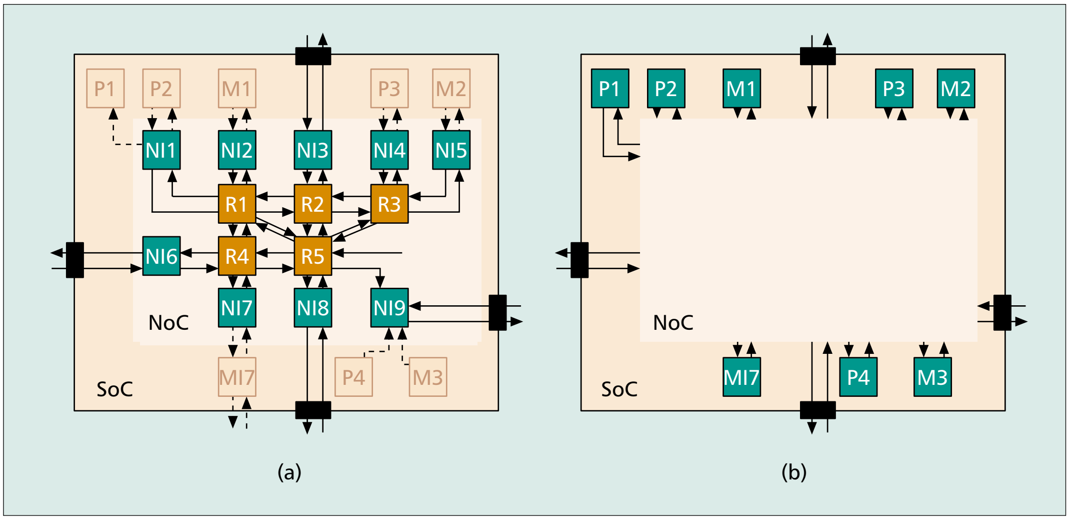


Figure 5. *Testing an SOC with a NOC.*

Silicon verification involves checking whether the hardware design is correct, assuming it has been manufactured correctly. This process is often referred to as silicon debug, as the goal is to try to find any remaining design errors that might have slipped through the pre-silicon verification stages.

to all routers at the same time. The responses can be compared to each other and any mismatches sent off-chip. This allows for a very efficient pass-fail decision on the NOC. Note that each block still needs to be made uniquely accessible for diagnostic purposes.

Timing tests are also very important for testing an NOC because:

- All clock boundaries between cores are inside the network interface.
- The NOC is spread over the entire SOC and therefore has many long wires.

These interconnection wires are more vulnerable to timing errors and crosstalk than others. Sending delay fault test patterns with high fault coverage over all communication links therefore increases the overall fault coverage. When NOCs become very large, additional test strategies such as those applied in field programmable gate arrays (FPGAs) can also be included, as described by Ubar *et al.* [7, Ch. 7].

Testing an SOC through Its NOC

After the NOC has been structurally tested, the network can be used to functionally transport test data to and from the cores in a very flexible way. Figure 5b shows how the functionality of the NOC is used during the test of the other blocks in the SOC. The NOC is now considered a known correct block that can be used to transport data from the device pins to the core under test and back. An advantage of reusing the NOC functionality is that no new TAM wires need be added to the design, as the existing NOC wires are reused. The flexibility of the NOC also enables the simultaneous distribution of test data to multiple identical cores. The test data itself can come from off chip or an internal BIST module.

When an SOC has structural errors, there are three possibilities:

1 The SOC is thrown away.
2 Redundant hardware present on the SOC can replace the faulty cores (repair).
3 The SOC is sold as a lower-performance SOC.

In cases 2 and 3, the NOC's flexibility is used to advantage. During the manufacturing test, the error information must be collected and subsequently permanently stored inside the SOC.

Silicon Verification

Silicon verification involves checking whether the *hardware design* is correct, assuming it has been manufactured correctly. This process is often referred to as *silicon debug*, as the goal is to try to find any remaining design errors that might have slipped through the pre-silicon verification stages. In this section we first discuss silicon debug of SOCs and communication networks. We then look in more detail at the options for verifying an NOC as part of an SOC, and how an NOC can be used for verification of the other cores of the SOC.

SOC Silicon Verification

Many companies have adopted design-for-debug strategies to allow prototype silicon to be efficiently and effectively verified [12–14]. The methods they use can be split into two complementary categories: those that change the configuration of the hardware of the chip to access debug data (intrusive) and those that can acquire debug data in parallel to the functional hardware (nonintrusive).

Intrusive debug: This category covers all debug methods that impact the application before debug data can be examined. These methods add on-chip breakpoint modules to the design of the SOC. These breakpoint modules interrupt the execution of the chip, after an internal sequence of events has been detected. All functional processing is at that point frozen. Various methods are then applied to access internal data. Commonly used access paths included system bus read and writes, TAP-based DMA access, and TAP-based scan-chain access [14].

Nonintrusive debug: This category covers those debug methods that allow examination of debug data in real time by streaming data for debug to an on-chip memory or out of the chip via a set of dedicated chip pins. Examples include the EJTAG and the IEEE-ISTO 5001 NEXUS standard. These methods add hardware to the design that only observes the functionality of the chip, operating completely in parallel to it. This allows the application to run at actual operating frequencies. As this debug architecture is completely separate (apart from probe points) from the functional hardware, care must be taken to keep the associated area cost within acceptable limits.

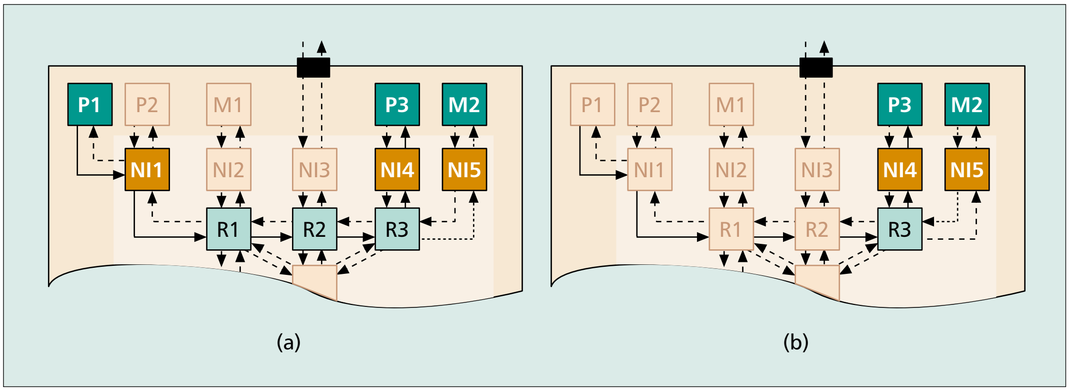
A hybrid solution is often chosen, combining these two methods, depending on the specific debug requirements.

Network Verification

In contrast to the previous section, network verification is about in-field testing. In the prior work on network verification two major areas can be recognized. First of all, a lot of work has been done on network errors: malfunctioning routers, routers that drop out of the network, links that are broken, error detection, and so on; and recovery procedures for all these cases. We assume the on-chip ÆTHEREAL network to be very stable, so network errors are not considered further.

The second major area of network verification focuses on bandwidth bottleneck detection and latency monitoring. This can be done either actively or passively. Active monitoring methods involve probing the network with test packets, in order to get round-trip latency, peak bandwidth, or available bandwidth. The problem with these methods is that they can introduce significant amounts of additional traffic in the network. On the Internet, this intrusiveness of debug traffic can easily be reduced by temporarily increasing the bandwidth with, for example, additional routers. Since this solution cannot be applied on an SOC, the intrusiveness cannot be removed easily, and thus complicates SOC debug.

Passive measurement methods execute performance measurements using special probe devices or probes added to routers, switches, or hosts. The measured data is cached; this cached



■ **Figure 6.** *Recording a stream for easy replay in the receiving processing unit: a) record events; b) replay events.*

Besides transporting functional data to a core via the NOC, it is also possible to transport other data. By recording the entire state of a core in an embedded memory, it is possible to quickly restore this state via the NOC prior to replay.

data can be either streamed to a central entity or shared within a local domain. How to best apply the lessons learned in computer networks to the ÆTHEREAL NOCs is an ongoing research activity within Philips.

VERIFYING THE NOC

In addition to using the standard verification techniques, an ÆTHEREAL NOC is verified using special events that have been defined inside the network blocks. With these events, conditions such as incorrect configuration, incorrect topology (e.g., two ports of the router are switched), incorrect initialization, and reset errors can be detected and used to verify applications.

Examples of events in the NI block are connection opened/closed, data for a connection received, data sent on a connection, and a certain data value appeared on a connection. Examples of events in the router block are data with a certain path is passing, data in a queue for more than *n* cycles, incorrect path, and conflicts in reservation. These events can be sent either actively (self-initiating) or passively (polling) Within ÆTHEREAL, the approach is to temporarily log events on-chip with a local timestamp, and later stream them off-chip for analysis. Special debugger software uses the event information to graphically represent the state of the network at different levels of detail. This provides the user of the debugger software with very useful debug data. Furthermore, the codebook approach [15] is applied to the off-chip data to correlate the generated events and isolate the root cause of a particular problem.

When deciding to use the on-chip network for transport events, a choice has to be made to make the implementation either completely independent of the NOC or to overdimension the existing network. Note that the latter approach leads to intrusiveness.

VERIFYING AN SOC THROUGH ITS NOC

Verification of the SOC contains two parts: verification of the NOC, described in the previous section, and verification of the cores. In general, access to the cores is a problem. However, when using an NOC this access is simplified. In Fig. 6 one of the possibilities is shown.

A processing block P3 receives data from another processing block P1 via the NOC. If, in the verification of P3, a situation has to be reproduced, it is necessary to also repeat P1 and its predecessors. If data from P1 to P3 is first tapped off and streamed to memory M2, P1 need not be executed during the repetition of P3 in following iterations. When, on replay, P3 repeats its fault, the data can easily be streamed off-chip and compared to the behavior of, for example, an FPGA model or a simulation model. Note that the exact timing of the data is lost during replay. This might lead to the disappearance of the problem or the introduction of a new problem. Nevertheless, the disappearance of the problem hints at a timing-related problem, which also aids in debugging the application. When the timing, captured at P3, is enforced via specific reservations in the NOC, it is even possible to eliminate this time intrusiveness. Experiments with varying timing behavior of the input data can also help to locate the problem.

Beside transporting functional data to a core via the NOC, it is also possible to transport other data. By recording the entire state of a core in embedded memory, it is possible to quickly restore this state via the NOC prior to replay.

In this section we have shown that the NOC introduces new options for locating a problem during verification of the cores. This type of verification is done only once per SOC design. In contrast, verification of an application that is mapped onto an SOC has to be conducted more often, and this is the topic of the next section.

APPLICATION VERIFICATION

Although the basic functionality of the SOC is verified using techniques from the previous sections, this by no means implies that the hardware and application software together will also run correctly. Examples of problems we might detect only after we have mapped an application onto the SOC are:

- A processing core that writes into another core's memory space and thereby corrupts its operation
- An incompatible format used to exchange data between cores (e.g., endianess)

- Deadlines not met because constraints were not passed to the network

These bugs will become more difficult to find due to the increasing complexity of the SOC itself. Many cores run in parallel, and the status of the system can no longer be related to a single program counter or traffic on a single bus. It is even possible that some of the cores execute multithreaded software, and these cores continue with those processes that can still consume or produce data. Lack of bandwidth (e.g., due to network congestion or functional processing spread) can cause an application to execute the processes in a different order. Some of the observed problems can be caused by use of the NOC. However, there are also new opportunities for verifying the complete application using the NOC (e.g., breakpoints, spying functional signals, and performance analysis).

Today and in the future, SOCs will be used to implement high-performance networking applications. One of the issues to be addressed for these SOCs is their on-chip data communication.

Breakpoints in the network can help analyze the state of the SOC in more detail, as breakpoints can stop (part of) the application by either gating (some of) the on-chip clocks or putting (part of) the cores in an idle mode. Once (part of) the application is stopped, the on-chip communication architecture can be used to access important registers and memories. Although in principle the ÆTHEREAL NOC has a global notion of time and can therefore be stopped in relation to this global clock, this is by no means trivial for NOCs in general. Furthermore, stopping an SOC is even more complex than stopping an NOC, as all cores can run at their own frequency. This is a topic of ongoing research within Philips.

Spying functional data in a local area network can be done relatively easy by plugging a network analyzer on that link. This network analyzer will monitor all data and process it into required debug or performance information. In an NOC, it is not possible to plug in a similar network analyzer because the data wires of the link are extremely difficult to probe. The two most important aspects we are interested in regarding a link are the possible congestion on the link, and the data from one of the connections that traverses the link. To achieve the first aspect, it is possible to add congestion monitors to the hardware design that can generate breakpoint events. To solve the second aspect, the NI can be configured such that the data is duplicated and sent along a separate debug channel. To reduce the tremendous amount of data generated during this functional spying, watchpoints can be introduced. Such a watchpoint only gives an indication if a certain value has passed and should be generated inside the NI because, in general, the routers have no knowledge about the data.

Performance analysis — Data and communication statistics, such as link utilization, latency, and jitter, are important when debugging an application. One way to gather statistics on latency is to let both the sending and receiving NI blocks generate a event. From the sending NI, the moment at w hich the data is written in its queue is valuable data. At the receiving NI, this is either the time it arrives, or the time the core retrieves it. The person debugging the application should decide which one is most useful. This technique is also very common in the verification of networks, as described earlier.

The latter two debug techniques lead to real-time generated data, which can be viewed as network events. These events are in addition to the network events defined earlier, and can be handled similarly.

CONCLUSIONS AND FUTURE WORK

Today and in the future, SOCs will be used to implement high-performance networking applications. One of the issues to be addressed for these SOCs is their on-chip data communication. In this article we presented the Philips ÆTHEREAL NOC for future-generation SOCs.

With the integration of a network on an SOC come additional test and verification requirements. Fortunately we can still use the wealth of test and verification methods that have already been successfully used in the past for either other existing SOCs and the much larger communication networks, such as LANs or the Internet. The integrated network also provides new and complementary possibilities to test and verify the SOC, and with it the SOC application. As shown in this article, there are plenty of options for meeting a particular SOC's test and verification requirements.

In the initial phase of the ÆTHEREAL project, key elements of future investigation were defined, which include the challenge of stopping an SOC when cores run on their own frequency. Due to dynamic runtime effects (voltage drops, crosstalk, alpha particles, etc.), and their growing size NOCs evolve to computer networks. How to apply the verification lessons learned in computer networks to NOCs in a cost-effective way is another interesting (research) challenge that is currently under investigation within Philips.

In the future, an NOC will most likely become a similar commodity as its bigger brother the Internet, and no doubt equally successful.

ACKNOWLEDGMENTS

The authors like to thank their colleagues Harald Vranken and Jos Huisken and the anonymous reviewers for their valuable feedback on an early draft of this article.

REFERENCES

[1] D. Wingard, "MicroNetworks-Based Integration for SOCs," *Design Automation Conf.*, 2001.

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Design Automation Conf.*, June 2001, pp. 684–89.

[3] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Comp.*, vol. 35, no. 1, 2002, pp. 70–80.

[4] D. Whelihan and H. Schmit, "Memory Optimization in Single Chip Network Switch Fabrics," *Design Automation Conf.*, June 2002.

[5] HyperChip Inc. Cell-Based Switch Fabric Architecture, World International Property Organization patent no. WO 02/098066 A2, Dec. 2002.

[6] F. Karim *et al.*, "On-Chip Communication Architecture for OC-768 Network Processors," *Design Automation Conf.*, June 2001.

[7] A. Jantsch and H. Tenhunen, Eds., *Networks on Chip*, Kluwer, 2003.

[8] E. Rijpkema *et al.*, "Trade Offs in the Design of a Router with Both Guaranteed and Best-effort Services for Networks on Chip," *Proc. Design Automation and Test Conf. in Europe*, Mar. 2003.

[9] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-switching Networks," *Proc. IEEE*, vol. 83, no. 10, Oct. 1995, pp. 1374–96.

[10] IEEE P1500: http://grouper.ieee.org/groups/1500/

[11] H. Vranken, F. Meister, and H.-J. Wunderlich, "Combining Deterministic Logic Bist with Test Point Insertion," *Proc Euro. Test Wksp.*, May 2002, pp. 105–10.

[12] D. D. Josephson, S. Poehlmann, and V. Govan, "Debug Methodology for the McKinley Processor," *Proc. IEEE Int'l. Test Conf.*, Baltimore, MD, Oct. 2001, pp. 451–60.

[13] H. Hao and R. Avra, "Structured Design-for-Debug — the SuperSPARC-II Methodology and Implementation," *Proc. IEEE Int'l. Test Conf.*, Washington, DC, Oct. 1995, pp. 175–83.

[14] B. Vermeulen, T. Waayers, and S. Goel, "Core-based Scan Architecture for Silicon Debug," *Proc. IEEE Int'l. Test Conf.*, Baltimore, MD, Oct. 2002, pp. 638–47.

[15] S. A. Yemini *et al.*, "High Speed and Robust Event Correlation," *IEEE Commun. Mag.*, May 1996, pp. 82–90.

Biographies

BART VERMEULEN (bart.vermeulen@philips.com) is senior scientist at Philips Research Laboratories, Eindhoven, The Netherlands. He received his M.Sc. degree in electrical engineering with honors from Eindhoven University of Technology, The Netherlands, in 1997. His research interests include the test and debug issues of large digital system chips.

JOHN DIELISSEN (John.Dielissen@philips.com) is a research scientist at Philips Research Laboratories, Eindhoven, The Netherlands. He received his M.Sc. degree in electrical engineering with honors from Eindhoven University of Technology, The Netherlands, in 2000. His research interests include on-chip communication in large digital system chips, with a current emphasis on networks on chip.

KEES GOOSSENS (Kees.Goossens@philips.com) received his Ph.D. from the University of Edinburgh in 1993 on hardware verification using embeddings of formal semantics of hardware description languages in proof systems. At Philips Research since 1995, he has worked on networks on chip for consumer electronics, where real-time performance, predictability, and costs are major constraints.

CALIN CIORDAS (C.Ciordas@tue.nl) is a junior researcher at the Technical University of Eindhoven (TUE), Design Methodology for Electronic Systems group. He obtained an M.Sc. in computer science from Technical University of Cluj Napoca, Romania, and a post-master technological designer degree in information and communication technology from Technical University of Eindhoven, Netherlands. His current research interest includes debugging and monitoring of on-chip multiprocessor systems and networks on chip.

In the future, a NOC will most likely become a similar commodity as its bigger brother the Internet, and no doubt equally successful.