NoC Monitoring: Impact on the Design Flow

Calin Ciordas
Electronic Systems Group
Eindhoven University of Technology
The Netherlands
c.ciordas@tue.nl

Kees Goossens
and Andrei Radulescu
Philips Research Labs
Eindhoven, The Netherlands
{kees.goossens,andrei.radulescu}@philips.com

Twan Basten
Electronic Systems Group
Eindhoven University of Technology
The Netherlands
a.a.basten@tue.nl

Abstract— **Networks-on-chip (NoCs) are a scalable interconnect solution to large scale multiprocessor systems on chip and are rapidly becoming reality. As the ratio of embedded cores per I/O pin increases, the run-time observability becomes a bottleneck. Run-time NoC monitoring can alleviate this problem. As NoCs are the result of sophisticated synthesis design flows, monitoring must be taken into account during this process. We present several scalable alternatives for NoC monitoring. The alternatives vary from using physically separated interconnects for user data and monitoring data, to a completely shared single interconnect. For each alternative we evaluate area cost, required design flow modifications, non-intrusiveness and reusability of monitoring resources for application communication traffic. An interesting trade-off is presented showing that what is area efficient requires efforts in modifying the NoC design flow and in achieving non-intrusiveness. All the experiments are done in the context of the Æthereal NoC and design flow.**

I. Introduction

Problem Description. Recent technological advances have increased the potential scale and complexity of future designs. This has led the designer's way to multiprocessor Systems-on-Chip (SoCs). Future designs of such large-scale chips need to be structured. Networks-on-Chip (NoCs) [1]–[6] have emerged as a scalable, future-proof SoC interconnect. They enable IP reuse and structuring of the design process by decoupling computation from communication and offering well defined interfaces.

As the complexity of each new SoC generation increases, the number of embedded cores and added functionality and features increases as well. With this added complexity getting the full design to work properly is increasingly difficult, requiring better system-level debug solutions. These in turn rely on run-time observability and controllability of the system. As current SoCs are incorporating more programmable cores, the controllability increases. The observability of such complex SoCs becomes a bottleneck as the ratio between the number of deeply embedded cores and I/O pins increases.

System level observability solutions must include on-chip instrumentation modules that support the entire system of interest, the computation and the communication part. Such instrumentation modules are common for computation, e.g. at the core level [7], and for bus-based communication [8]. As the SoC world is moving towards NoCs, inter-IP communication will be able to use multiple truly parallel communication paths, as opposed to centralized bus communication in current SoCs. While a central bus monitor like [8] is enough for bus-based systems, multiple monitoring probes are required in NoC-based SoCs [9] to keep up with the communication parallelism. Supporting NoC observability in future systems is a must.

The necessity of multiple probes leads to the problem of their interconnection. Interconnecting the chip-wide distributed monitoring probes poses a significant challenge. Any such interconnect must be scalable, non-intrusive (which is a key aspect in debugging, one of the main run-time monitoring drivers), run-time usable and configurable, and of minimum area cost. As NoCs are a scalable interconnect they appear naturally fit for the task [9]. Options like sharing or not sharing a single interconnect for functional and monitoring traffic are key to monitoring system design. However, NoCs are the result of sophisticated NoC synthesis design flows. Some monitoring probe communication requirements are not known beforehand, but only after the NoC to be probed has been designed, or at least some steps in the NoC design flow have been performed. For example, some requirements may be known only after topology generation (such as the number of routers employed in the NoC, which is relevant if all routers need to be probed), other after the mapping or path selection. In this case the monitoring problem must be solved within or at least coupled with the NoC design process having an impact on the overall NoC design flow.

Related Work. There has been a lot of work towards run-time observability and towards NoCs, but little on the combination of the two. ARM's Coresight [10] technology combines ETMs [7] for ARM cores, with the AHB Trace Macrocell which gives visibility on AMBA AHB busses. First Silicon's on-chip instrumentation technology (OCI), provides on-chip logic analyzers [8] for AMBA AHB, OCP, and Sonics SiliconBackplane bus systems. These allow the user to run-time capture bus activity, and in a multi-core embedded debug system [11] they can be combined with in-system analyzers for cores, e.g. for MIPS cores. Although state-of-the-art, both solutions are not able to cope with a NoC-based SoC. The test and verification implications of using NoCs have been inventoried in [12]. Currently, in the NoC research community, focus is on the design [2], [4], [5], [13]–[15], analysis [16], [17] and use [18], [19] of NoCs. [9] proposes a generic NoC monitoring service (NoCMS) consisting of hardware probes attached to NoC components, routers and network interfaces (NIs). It assumes that the probes connect to the NoC NIs and use the existing physical NoC for monitoring data transport.

Contribution. This paper presents several monitoring alternatives. These are independent of any specific NoCs. All options are based on the reuse of NoC components and (parts of) the NoC design flow. As all are based on a NoC interconnect, they are all scalable solutions. For each of the proposed solutions, we explain the main concepts and the architectural details. We evaluate all the proposed solutions with respect to four aspects: (1) impact on the overall NoC design flow, (2) non-intrusiveness, (3) area cost and (4) reuse potential of debug resources. All our options are proven and exemplified with the Æthereal NoC and design flow.

II. NoC Monitoring Service and Æthereal NoC

The Æthereal NoC [3], [15] runs at 500 MHz and offers a raw link bandwidth of 2GB/s in a 0.13μm CMOS technology. Æthereal offers transport-layer communication services to IPs, in the form of connections, comprising best-effort (BE) and guaranteed-throughput (GT) services. Guarantees are obtained by means of TDMA slot reservations in NIs. Æthereal NoC instances are reconfigurable at run-time.

The NoCMS [9] consists of hardware probes (Ps in Figure 1) attached to NoC components, routers and NIs. The generic probe architecture includes a sniffer for data gathering from the NoC components, an event generator for processing of this data, and a monitoring network interface to send the monitoring data to a NoC external device, the monitoring service access point (MSA). This is done via the NoC by means of GT connections. Specialized monitoring probes may be integrated in the NoCMS, e.g. probes able to trace data flits in the NoC at run-time, or able to follow data traffic to compute averages and statistics of it. The NoCMS also provides support for the chip-wide monitoring system by offering the option of integrating third party probes (monitoring IPs) like ARMs ETM probe [7] for the ARM processors. Adding a NoCMS to an existing NoC means: (1) adding the probes (the number of probes may depend on the NoC topology, mapping of cores to NIs, number of NIs connected to routers), (2) adding one MSA assuming a centralized monitoring system and (3) connecting the probes to the MSA for the purpose of data transport and run-time configuration.

III. Monitoring Interconnect Options

In this section we consider and explore three monitoring platform options:

(A) Separate Physical Interconnect for the original NoC application and the NoCMS
(B) Common Physical Interconnect but Separate Physical NoC Resources
(C) Common Physical Interconnect and Shared Physical NoC Resources

All these three options are supported in the Æthereal design flow. As a reference example we have chosen an MPEG codec with a 2x3 mesh NoC interconnect [15]. The area cost of the NoC interconnect is 2.35mm^2. In the following, for each option, we explain the concepts, the impact on the NoC design flow, the non-intrusiveness aspect, the reuse potential of debug resources for application traffic, and the interconnect area cost, not including the area of the probes which is the same in all cases. Each of the options is compared to the original NoC, called user NoC in the remainder, shown in Figure 1(a). The typical NoC design flow [13], [15], [19] is normally split in four steps as shown in Figure 2(a): topology selection, mapping, path selection and slot allocation. Each step adheres to the decisions taken in the previous steps. As prerequisites for NoC design, communication requirements must be derived, and the set of IPs to be connected to the NoC must be specified. In the topology selection step, the router network together with the bordering NIs are generated, based on the previously derived communication requirements. Using this topology together with the IP specification, the binding of IP ports to NI ports is done in the mapping step. In the path selection step, paths are allocated for all the communication flows specified, and in the slot allocation step each of the flows gets its own TDMA time slots for the traversed NoC links. Some design flows may omit or combine various steps.

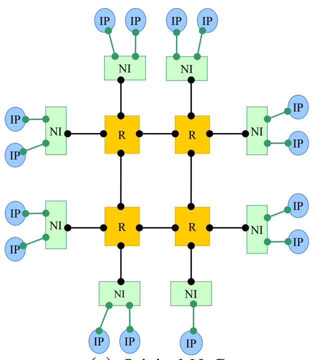
A. *Separate Physical Interconnect*

In this case a separate physical interconnect is chosen for monitoring. Although any interconnect may be used, we have chosen to use a NoC, the monitoring NoC, because it is scalable. Figure 1(b) show the resulting system. The monitoring NoC is used for transporting the monitoring data from probes to the MSA and for monitoring configuration traffic from MSA to the probes. The monitoring NoC can be similar in topology with the user NoC interconnect. For simplicity, we only show a fully probed NoC in Figure 1(b), with probes attached to all routers. A more advanced, selective NoC probe placement at routers is possible, e.g. ensuring a coverage of all NoC physical links. In the remainder only a fully probed NoC is assumed as well. For each of the probed routers we add a new router and an NI. The NI is used by the probe to connect to the monitoring NoC. Please note that probes can be attached also to NIs or IPs in the system, in which case these will connect to the monitoring router corresponding to the user router these NIs or IPs connect to in the user NoC. Each of the probes and the MSA connect to the monitoring NoC through a separate NI. Optionally, taking into account the monitoring requirements driven e.g. by debugging, some of the monitoring NoC links (in between routers) may be removed, as long as each probe can still connect to the MSA.

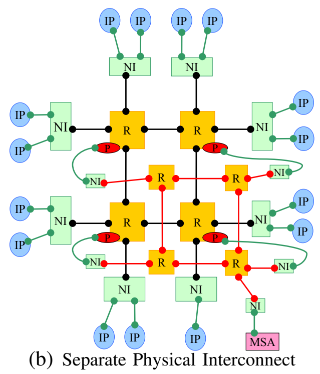
Design Flow Impact: During the NoC design process, the NoC design flow is applied twice: (1) for the user NoC, taking into account the user communication requirements as shown in Figure 2(a), (2) for the monitoring NoC, taking into account the monitoring communication requirements as shown in Figure 2(b). Dimensioning of the monitoring communication requirements and of the number of debug IPs (e.g. router probes) required, which are dependent on the user NoC topology, mapping, and path selection, is simple as all these aspects for the user NoC are not influenced in any way by the monitoring system and done beforehand. While applying the NoC design flow for the monitoring NoC, topology is already given by the original NoC, and mapping is given by the probe placement in the original NoC, as previously explained. Therefore only the path selection and slot allocation have to be done for the monitoring NoC.

Non-intrusiveness: This solution is non-intrusive because only the monitoring NoC is used for transporting the monitoring data. No interference between monitoring NoC and user NoC is possible, because they are physically disjoint.

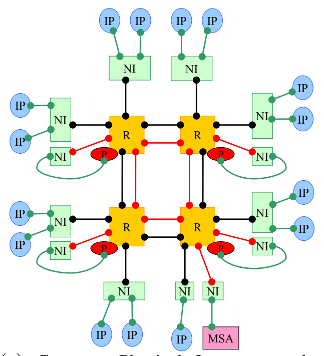
Area cost: A total NoC area cost of 3.82mm^2 (2.35mm^2 original + 1.47mm^2 extra) was determined based on the



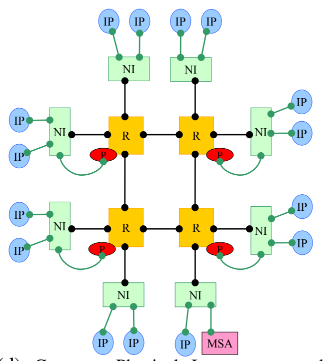
(a) Original NoC



(b) Separate Physical Interconnect

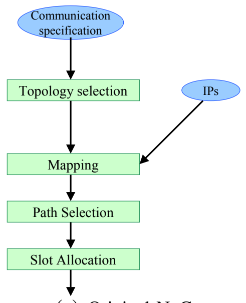


(c) Common Physical Interconnect but Separate Physical Resources

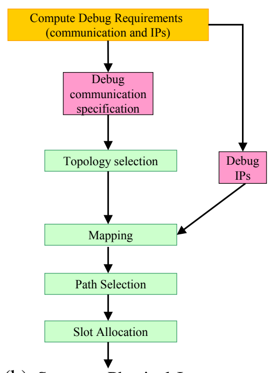


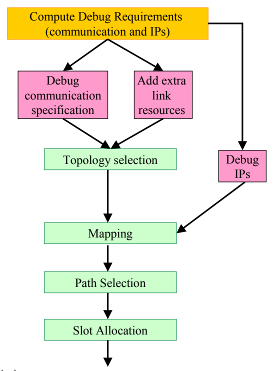
(d) Common Physical Interconnect and Shared Physical Resources

Fig. 1. Monitoring Transport Options



(a) Original NoC


Compute Debug Requirements (communication and IPs)
Debug communication specification
Topology selection
Debug IPs
Mapping
Path Selection
Slot Allocation
(b) Separate Physical Interconnect



(c) Common Physical Interconnect but Separate Physical Resources

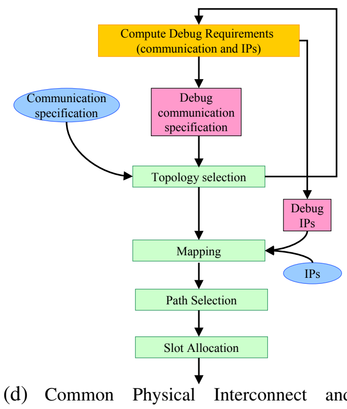

Compute Debug Requirements (communication and IPs)
Communication specification
Debug communication specification
Topology selection
Debug IPs
Mapping
IPs
Path Selection
Slot Allocation
(d) Common Physical Interconnect and Shared Physical Resources

Fig. 2. Design Flow Modifications

addition of 7 NIs for the 6 probes and one MSA, and of 6 routers.

Reuse: No reuse potential due to complete separation of monitoring and user NoCs.

B. Common Physical Interconnect but Separate Physical NoC Resources

An alternative monitoring option is to have within the user NoC a separate monitoring subnetwork. No new routers are added, but following the NoC topology, separate links and their corresponding router ports are added to the existing NoC. Each probe and the MSA gets its own NI to connect to the NoC. This is visually depicted in Figure 1(c).

Design Flow Impact: During the NoC design process the NoC design flow is applied twice: Considering the user requirements, the user NoC is obtained, by going through the NoC design flow. In this way, the topology, mapping, path selection and slot allocation are computed for the user NoC as shown in the reference design flow of Figure 2(a). At the second run of the design flow, as shown in Figure 2(c), the debug communication requirements and the required debug IPs, e.g. one probe for every router in the original NoC, are derived. This is done based on the user NoC topology. The user NoC is then extended with the monitoring resources, router links. In the path selection and slot allocation steps, the newly added router links are only scheduled for the monitoring traffic. Optionally, taking into account the monitoring requirements, driven e.g. by debugging, some of the monitoring router ports and links may be removed from the monitoring subnetwork, as long as each probe can still connect to the MSA. Dimensioning of the monitoring communication requirements which are dependent on the user NoC topology, mapping, or path selection is simple as the user NoC path selection and scheduling is not influenced in any way and done beforehand.

Non-intrusiveness: This solution is non-intrusive as only the monitoring subnetwork consisting of dedicated links is used for transporting the monitoring data. Although the routers are shared, the set of user links and the set of monitoring links are disjoint. No interference between the monitoring subnetwork and the user subnetwork is therefore possible. Existing user NoC scheduling in the original NoC (Figure 1(a)) is kept also in the new NoC (Figure 1(c)).

Area cost: This solution has a high NoC area cost: 3.88mm^2 (2.35mm^2 original NoC + 1.53mm^2 extra). This was due to increasing the arity of all six routers, e.g from 3 to 6, and the addition of 7 separate NIs, from which 6 for the probes and one for the MSA.

Reuse: One advantage is that after debugging, some debug communication resources (the set of monitoring links) can be used partially or totally for functional user traffic.

Disadvantage: One potential disadvantage of this solution is that the routers are limited to a maximum number of ports.

C. Common Physical Interconnect and Shared Physical NoC Resources

A third possibility is to use the existing user NoC for the user traffic and also for the monitoring traffic. Both would share all the NoC resources but we keep the NoC user traffic and the monitoring traffic separated. In this way a virtual NoC for monitoring is created.

Design Flow Impact: Considering the user requirements, the user NoC is obtained, by going through the reference NoC design flow from Figure 2(a). In this way the topology and mapping are computed. After this, the monitoring communication requirements and debug IPs are computed and probes are added to the design. Figure 1(d) shows that probes are connected to the existing NoC by means of an extra port on the existing user NIs, as opposed to separate NIs for monitoring in the previous two cases. All the links, NI and router links, are considered shared between the user and the monitoring traffic. The mapping of the probes to existing NIs is based on the closest available NI. Path selection and slot allocation is computed together for all the communication requirements: user and monitoring. There are two possible cases:

(1) Everything fits on the existing user NoC. This means that the user NoC can accommodate the monitoring communication requirements on top of the existing user communication requirements. Topology of the NoC will therefore not change. This is exactly the situation shown in Figure 1(d). In this case, we have the lowest area cost, as no new NoC components, routers and NIs for monitoring, are added, except the new NI ports to connect the probes to the NoC.

(2) It does not fit on the existing user NoC. In this case, a new NoC must be generated, e.g. by increasing the topology and repeating the process. By increasing the topology, the number of user NoC routers increases and in turn the number of required monitoring probes may increase as well (e.g. if probing all routers is required). This leads to the recomputing of the monitoring communication requirements and monitoring IPs as shown in Figure 2(d). However, this process may not converge, i.e. a solution may not be found.

Non-intrusiveness: By sharing NoC resources, non-intrusiveness is potentially not guaranteed and must be enforced. The monitoring traffic can in this case interfere with the user traffic. For our Æthereal examples this was not needed because we use GT for both functional and monitoring traffic and they cannot interfere. However, in general extra steps may be required in order to enforce the non-intrusiveness.

Area cost: The total NoC area cost for our example is 2.75mm^2 (2.35mm^2 original + 0.4mm^2 extra). This was based on the addition of 7 network interface ports, 6 for connecting the probes and 1 for the MSA. The added monitoring traffic fits in the original network.

Reuse: After debugging, the debug communication resources can be used for functional user traffic, e.g. by BE traffic.

A brief overview, summarizing the advantages and disadvantages, of each of the proposed solutions is shown in Table I. A, B and C are the solutions proposed in the Sections III-A, III-B and III-C respectively.

The table shows that having separate NoCs or NoC resources just for monitoring, as A and B, is both non-intrusive and basically straightforward in the NoC design flow; however it shows a high area cost in both cases. Having shared resources for user traffic as well as for monitoring traffic is a good idea area-wise but may have strong implications on both the NoC design flow and the non-intrusiveness. However, both of these can be alleviated, as previously explained. Furthermore, it enables reuse of the shared resources by the functional traffic after the debugging is done.

TABLE I
COMPARISON

	A	B	C
Design Flow	++	+	-
Non-intrusiveness	+	+	+/-
Area Cost	-	-	+
Reuse after debugging	-	+	+

IV. CONCLUSION

We have presented three architectural options for a NoC monitoring service supporting a chip-wide monitoring system. All options are generic and can be applied to any NoC. Each of the presented options is NoC-based and scalable. Non-intrusiveness, influences on the overall NoC design flow, area, and reuse potential are evaluated for all these options. An interesting trade-off is presented showing that what is good for area and reusability requires efforts in modifying the NoC design flow and in preserving the non-intrusiveness of the monitoring system.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–80, 2002.
[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. DAC*, 2001, pp. 684–689.
[3] K. Goossens *et al.*, "Networks on silicon: Combining best-effort and guaranteed services," in *Proc. DATE*, Mar. 2002, pp. 423–425.
[4] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proc. DATE*, 2000, pp. 250–256.
[5] M. Millberg *et al.*, "The Nostrum backbone - a communication protocol stack for networks on chip," in *Proc. Int'l Conference on VLSI Design*, 2004, pp. 693–696.
[6] F. Karim *et al.*, "An interconnect architecture for networking systems on chips," *IEEE Micro*, vol. 22, no. 5, pp. 36–45, Sept. 2002.
[7] ARM, *Embedded Trace Macrocell*, www.arm.com.
[8] F. Silicon, *BusNavigator*, http://www.fs2.com/busnavigator.html.
[9] C. Ciordas *et al.*, "An event-based monitoring service for networks on chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 4, pp. 702–723, Oct. 2005.
[10] ARM, *Coresight*, http://www.arm.com/products/solutions/CoreSight.html.
[11] R. Leatherman and N. Stollon, "An embedded debugging architecture for SoCs," *IEEE Potentials*, pp. 12–16.
[12] B. Vermeulen *et al.*, "Bringing communication networks on chip: Test and verification implications," *IEEE Communications Magazine*, vol. 41, no. 9, pp. 74–81, Sept. 2003.
[13] E. Bolotin *et al.*, "QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture*, vol. 50, no. 2–3, Feb. 2004.
[14] S. Kumar *et al.*, "A network on chip architecture and design methodology," in *Proc. Symposium on VLSI*, 2002.
[15] K. Goossens *et al.*, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proc. DATE*, Mar. 2005, pp. 1182–1187.
[16] G. Pestana *et al.*, "Cost-performance trade-offs in networks on chip: A simulation-based approach," in *Proc. DATE*, 2004, pp. 764–769.
[17] P. Poplavko *et al.*, "Task-level timing models for guaranteed performance in multiprocessor networks-on-chip," in *Proc. CASES*, 2003, pp. 63–72.
[18] K. Goossens *et al.*, "Interconnect and memory organization in SOCs for advanced set-top boxes and TV — evolution, analysis, and trends," in *Interconnect-Centric Design for Advanced SoC and NoC*, J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Eds. Kluwer, Apr. 2004, ch. 15, pp. 399–423.
[19] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. DATE*, 2004, pp. 896–901.