Networks on Chips for High-End Consumer-Electronics TV System Architectures

Frits Steenhof[1], Harry Duque[2], Björn Nilsson[2], Kees Goossens[3], Rafael Peset Llopis[1]
[1] IC Lab, Philips Consumer Electronics, Eindhoven, The Netherlands
[2] Department of Information Technology, University of Lund, Sweden
[3] Philips Research Laboratories, Eindhoven, The Netherlands
{frits.steenhof,kees.goossens}@philips.com

Abstract

Consumer electronics products, such as high-end (digital) TVs, contain complex systems on chip (SOC) that offer high computational performance at low cost. Traditionally, these SOCs are application-specific standard products (ASSPs) with limited programmability. We describe why TV SOCs must become more flexible, and why companion chips together with networks on chips (NOC) are a crucial enabling technology. In particular, networks that span multiple chips will become important in the near future.

We demonstrate our ideas by extending a commercially-available SOC for picture improvement in high-end TVs with the Æthereal NOC. Our first unoptimised results indicate that replacing the original interconnect (consisting of dedicated links and multiplexers for bypasses) by programmable NOC increases the SOC area by 4% and its power dissipation by 12%. The new, flexible SOC allows new tasks to be spliced in at any point in the task graph. Both analytical performance verification and system simulations at RTL VHDL show that the extended SOC meets its functional requirements. Using the Æthereal design flow the extended architecture was designed, implemented, and verified in 12 person months.

To the best of our knowledge, this is the first application of a NOC to a commercial SOC. The quantitive results indicate that even retrofitting a NOC to an existing architecture is beneficial at acceptable cost.

1 Introduction

Televisions (TV) are consumer-electronics devices that are split in three segments: low end, mid end, and high end.[1] The focus for low and mid end TVs is primarily on low cost. The ICs in these products are produced in high volumes, and their functionality is highly tuned. Hence single application-specific standard products (ASSPs) are the preferred implementation. The situation for high-end TVs significantly differs, however. To be leading in the market as a whole, *branding* is important, and it is imperative to be present in the high-end market segment. High-end products serve as demonstrators for the latest technology and highest quality. Thus, innovation first takes place in the high-end segment, and after some time the mid and low-end follow. Hence, the focus is on a *high innovation rate and high performance*. Price margins in the high-end segment are considerably higher than in the low and mid-end. As a result, although smaller in terms of volume, the turn-over of the high-end segment is comparable to half of the mid-end segment.

High-end TVs are differentiated by picture quality. Many advanced techniques are used, such as [9]:

- standard-definition and high-definition pictures: horizontal and vertical scaling;
- higher picture rates: 100Hz with digital natural motion;
- multiple windows: picture in picture, split windows;
- contrast improvement: sub-pixel-based luminance transient improvement and dynamic contrast;
- colour improvements: colour-dependent sharpness, green enhancement, blue stretch, skin-tone correction;
- 3-dimensional digital noise reduction.

A high-end TV contains megabytes of embedded software, and easily performs 400 GOPS [13, 20].

Companion chips

The SOCs for all market segments are based on the Philips's Nexperia Home platform [7]. In fact, all but the simplest TVs use *multiple chips* (or dies: system in a package), for which there are several reasons.

First, external (DDR) memory chips are used to store the large amounts of data (e.g. video frames for temporal upconversion, noise reduction, and de-interlacing [9, 13, 20]).

Second, TV SOCs are sold by Philips Semiconductors to customers other than Philips Consumer Electronics. To ensure that this does not conflict with the goal of differentiation for the high-end segment, only a subset of TV functionality is offered to non-Philips customers during a certain lead time. In other words, new differentiating features are not sold to competitors for a certain period. Conversely, competitors may have proprietary functionality for differentiation of their TV sets, which they do not wish to be integrated in a Philips solution. This leads to the concept

[1] Although we focus on TVs in this paper, the arguments presented here hold for many other consumer-electronics devices, such as mobile phones and set-top boxes.

3-9810801-0-6/DATE06 © 2006 EDAA

of a *companion chip* that encapsulates differentiating functionality. Companion chips are usually made by Philips Consumer Electronics, and are not available to competitors. The common, non-differentiating functionality can be sold as a separate chip to other customers. The larger production volume yields a lower price for the common functionality.

Third, companion chips are used to manage the *different innovation rates* in the different market segments (low in low end, and high in high end). To maximise innovation (and minimise time to market) in the high-end segment, new functionality is implemented in preferably-programmable, lower-volume companion chips. Later, this functionality is optimised and merged with the high-volume main chips, to minimise cost. This so-called waterfall model is used to manage the migration of functionality from the high-end to the mid and low-end SOCs.

Fourth, companion chips provide *more flexibility* than programmable components in a single SOC. Flexibility can be achieved in a SOC by including programmable components such as SIMD [1] and VLIW [22, 24] processors, and embedded FPGA [6]. However, in all cases the computation and/or storage capacity of these components is fixed when the SOC is designed. With companion chips, this decision is taken later, when more is known about the added functionality, and a better (i.e. cheaper) fit can be achieved. Moreover, programmable ASSP solutions do not offer sufficient computational power for the latest high-end functionality implemented in companion chips, which tend to be ASICs.

Finally, companion chips reduce *development risk*. Different existing or potential new functions will in general have different requirements. With a single SOC, a superset of all requirements must be instantiated [20], whereas otherwise multiple specialised companion chips can be used. The latter also reduces the significant risk of developing a single very complex SOC.

In this paper, we aim to prove that *NOCs are a mature technology that can be applied beneficially even to existing SOC architectures*. In particular, retrofitting a NOC in a commercially-available SOC for picture improvement in high-end TVs, makes that SOC more flexible and re-usable. Using NOCs across multiple chips (e.g. a main chip with companion chips) has additional advantages, as described above.

In the remainder of this paper, we first describe a particular companion chip for high-end TVs, and how it can be used in combination with PNX8550 (also known as Viper2) (Section 2). In Section 3 we explain why NOCs [3] are a key enabling technology for building systems containing multiple (companion) chips. A new SOC architecture for the companion chip, containing a NOC is defined in Section 4. Section 5 quantitively compares the original and extended architectures. Section 6 concludes.

2 Current TV System Architectures

Figure 1 shows an example TV system, with a main TV chip PNX8550 [13, 21] and a companion chip, each with an external memory. (Multiple companion chips are also possible.) The main SOC is the master of the system; it interacts with the user, TV source, TV display, and peripherals, and it configures the companion chip(s). The main SOC contains some 60 IP blocks, and the companion chip (large) 9 IP blocks. The main chip and companion chip communicate using a *high-speed external link* (HSEL), which can be implemented with a proprietary or standard link-level interconnect technology, such as PCIExpress [19]. (The connections to the external memories use the standard DDR(II) interface.) Using the HSEL, the companion chip(s) can use the external memory of the main chip, in addition to their own external memory.

Functionality of the system is usually defined as a collection of hundreds of *task graphs* (also called modes or use cases). There are several reasons for the high number of task graphs.

First, the diversity in I/O: some 100 source formats (e.g. VGA, ATSC, DVB, DAVIC, analogue) and around 50 display types have different characteristics such as lines, pixels, frequency, and mode (standard, film).

Second, based on source, display, and user preferences, different picture-improvement functionalities, and various GUI modes (picture in picture, split window, etc.) can be combined.

The third reason is that traditionally a single SOC is defined that implements all task graphs for all possible applications of the SOC ("superset approach" [20]). This maximises the production volume, and hence lowers the cost. However, this results in a complex single-SOC solution. The companion-chip approach splits the functionality over multiple simpler SOCs. The system is then composed by mixing and matching a main SOC with (multiple) different companion chips. Because task graphs will span all chips it is essential that the composition of main chip and companion chip(s) is easily programmable. The dashed line in Figure 1 shows a task graph consisting of 11 IP blocks that spans a main chip, companion chip, and two memories. Networks on chips will play an important role here, as we shall see.

In the following section we discuss how companion chips and NOCs together can reduce the complexity of the system solution.

3 The role of NOCs for TV SOC Architectures

In the introduction we identified the trend to partition system functionality over multiple chips or dies (system in a package). Ultimately, however, the multiple chips implement a single system. Hence it is important to be able to design a system as a whole, and then to be able to seamlessly distribute it over multiple chips. Alternatively, a given combination of (companion) chips should be easily combined to a working system.

Networks on a chip [3–5, 11] promise to ease the design of complex SOCs because they: (a) structure and manage wires in deep submicron technologies, (b) use wires efficiently through sharing, (c) scale better than busses, (d) are programmable for multiple and new task graphs, and (e) decouple computation from communication through well-defined interfaces, enabling IP blocks and interconnect to be designed in isolation, and to be integrated more easily.

The rationale for networks on a chip has been the need for scalable high-performance interconnects for SOCs, but until now, their scope has been limited to a single chip. We propose that the *scope of NOCs should be extended to transparently span multiple chips.*

Networks on different chips are connected through high-speed external links (HSEL). There are a number of issues to be resolved, such as different characteristics of on-chip and off-chip links perhaps requiring bridges, end-to-end flow control, addressing, and routing. Basic solutions for all these issues have been identified for the Æthereal NOC.

As a result, we obtain the flexibility to easily combine multiple (companion) chips to a single system, i.e. compose task graphs on

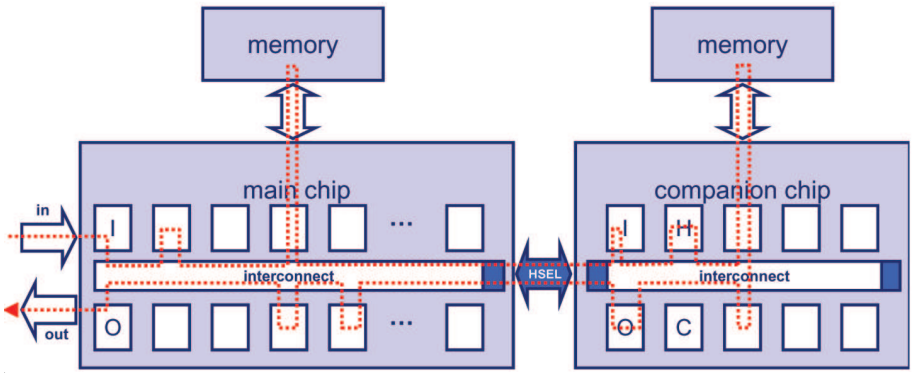

Figure 1. The main TV chip, companion chip, and external memories.

different chips to a single task graph. In particular, when a companion chip is an FPGA, new systems can be built very quickly. If the FPGA is configured with a number of IP blocks interconnected by a NOC [15,18] (routers, network interfaces, and perhaps a HSEL bridge as suggested by the dark boxes in Figure 1), then low-cost main chips implemented as an ASIC or ASSP can be easily and quickly combined with the latest features implemented in FPGA.

Currently, systems are built from several companion chips that are combined in an ad hoc manner. NOCs should make this process more structured, simpler, and faster.

To illustrate and quantify the role of NOCs in companion chips, we modified an existing TV companion chip by replacing its dedicated interconnect by a NOC with a port dedicated to a HSEL. Section 4 describes the original and modified architectures, which are compared quantitively in Section 5.

4 TV SOC Architecture with NOC

In this section we briefly describe the architecture of a commercially-available TV companion chip, and a version of the SOC extended with a NOC.

4.1 Original architecture

The TV companion chip contains 9 IP blocks for enhancing video (some IP are large subsystems). Their processing order is fixed, except for one IP block (horizontal scaler H), which can appear at one of two places in the processing pipeline. The interconnect for the SOC consist of dedicated links (wires), and some bypass logic (multiplexers) for the horizontal scaler. Video data enters and leaves the companion chip through a single HSEL. Depending on the mode and location in the processing chain, pixel data is 8, 9, or 10 bits in 4:4:4, 4:2:2, or 4:2:0 Y:U:V format. Pixel data is augmented with 4 bits of side-band information (e.g. end of field). All IP blocks use a streaming data protocol (DTL peer-to-peer streaming data [23]).

The *goal of enhancing the SOC* with a NOC is to improve its flexibility and scope for re-use. For example, an improved version of an algorithm implemented by an IP block may have become available, and we would like to upgrade it. Or, a new function becomes available, and we would like to insert it in the processing pipeline, which may require changing the order of IP blocks.

4.2 New architecture

To address these issues, we should replace the dedicated interconnects of the main chip and the companion chip by a NOC. We have sketched how the former could be done in [13]. In this paper we take one more step, and replace the dedicated interconnect of the companion chip by an Æthereal NOC [12, 25], as shown in Figure 2. The NOC is shown on a shaded background, with its components (routers R, network interface kernels K, and network interface shells S). The number of master (M) and slave (S) ports are shown at each NI. The NOC enables a programmable order of IP blocks. The SOC's input and output are shown by the HSELIO block, which directly connects to two processing blocks, for legacy reasons. A new HSEL to attach another companion chip, such as an FPGA, is directly connected to two master and two slave NI ports. The IP blocks include a horizontal scaler (H), two new blocks (NB1, NB2), and a control processor (C), which will be discussed below.

The dashed line in Figure 2 illustrates a task graph that includes a new block (NB2) on another companion chip.

4.3 Multiple data formats

A particular IP block may, in different modes, receive or send pixel data in different formats, but a network interface (NI) kernel port has a fixed width (e.g. 32 bits). The function of the NI shell [25] is to efficiently pack 8, 9, or 10 bit pixels in 4:4:4, 4:2:2, or 4:2:0 formats in multiple words of 32 bits, depending on the mode. In the original architecture 4 bits of side-band information were sent with every pixel, even though the information changed only very infrequently. With dedicated point-to-point wires, this is optimal in terms of bandwidth and power dissipation. For a NOC, however, highly repetitive information wastes shared bandwidth and dissipates power. Hence, in the new architecture side-band information is sent only when it changes. A 32-bit word therefore contains a 1-bit header, identifying it as a data message or a control message. The former consists of three pixels (of 8, 9, or 10 bits), and the latter of three 4-bit side-band information items.

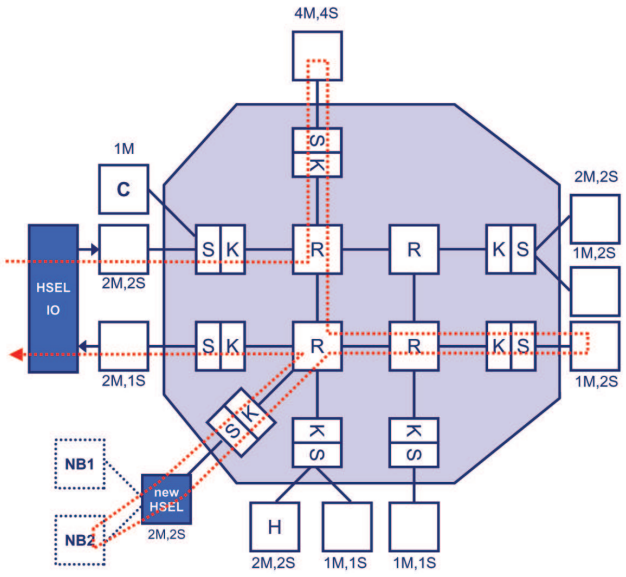

Figure 2. Architecture of companion chip with network on chip.

In this way the volume of side-band information is reduced from 12%-33% in the original SOC to 1% in the new version. A simple multi-format NI shell was implemented, to pack 4:4:4, 4:2:2, 4:2:0 pixels in 3, 2, and 1 words respectively. This shows that decoupling of protocol-specific functionality (in the NI shell) and the protocol-independent functionality (in the unchanged NI kernel) works well in practice [25].

4.4 Task graphs

The companion chip implements hundreds of hard-real-time task graphs. The NOC must support all of them, with a guaranteed bandwidth and latency. At the time this experiment was performed, automatic NOC dimensioning based on multiple task graphs was not yet supported by the Æthereal NOC design flow [11]. A NOC instance and IP mapping was therefore specified by hand.[2] However, configuring a given NOC with multiple task graphs is automated, and entails buffer sizing, path finding, TDMA slot allocation, etc. [14, 25]. For all task graphs, the performance verification tool in the Æthereal design flow analytically verifies that the NOC hardware and NOC configuration correctly implement the specified task graph [10]

Recall that the reason to insert a NOC in the companion chip is to be able to replace existing IP blocks with new functionality, or to insert new functionality at any point in the task graph. New functionality is implemented on another companion chip, communicating using a new HSEL. For example, two new IP blocks (NB1 and NB2 in Figure 2) on the new companion chip with a master and slave port each, can be inserted anywhere in the task graph. (An example is given by the dashed line in Figure 2.) The task graphs were modified to reflect this requirement. The worst-case task graph contains 17 connections with a cumulative bandwidth requirement of 2 GByte/sec. The mapping of IP blocks to NIs balances the bandwidth over the NOC to reduce contention, and thus the number of TDMA slots, buffer sizes, and latency.

[2]Multi-task-graph NOC mapping and configuration have been automated since this work [14, 16].

4.5 NOC

As a result of the (verified) NOC dimensioning and configuration, a 2×2 mesh with routers of arity 3, 4, and 5 is used. The TDMA slot table contains 20 slots. A total of 10 IP blocks are connected to the 8 NIs, using 38 NI ports with one connection each. One additional port per NI (8 total) is used for programming the NOC [25]. The number of master (M) and slave (S) IP ports of each NI is shown in Figure 2.

A new NI shell architecture (see 4.3) was designed and implemented. The RTL VHDL of the complete NOC (routers, NI kernels, and NI shells) is automatically generated by the Æthereal design flow. The NOC was integrated with the unmodified RTL of the IP blocks of the original companion chip.

4.6 Configuration and simulation

The Æthereal NOC is (re)configured at run time with the required task graph using memory-mapped IO [12]. We modelled a control processor that is directly connected to the NOC (block C in Figure 2). In a definitive implementation the MIPS control processor on the main chip [13] would run the embedded reconfiguration software. This is not an issue, because the NOC is programmed using the NOC itself as a control interconnect, and the location of the control processor in the NOC is not important. Moving the reconfiguration software to another processor requires no changes to either NOC or embedded software.

To simulate the entire original system (main chip PNX8550, companion chip, and two DDR memories), the RTL VHDL description of the companion chip was used, together with a high-level C model of the main chip. The same set-up was used for the modified companion chip, i.e. original RTL VHDL of IP blocks and RTL VHDL of the NOC were simulated with a high-level model of the main chip. Both simulation and analytical performance verification proved the new architecture functionally correct.

In the next section we compare original and extended companion chip architectures quantitively.

5 Results

The original companion chip uses wires for direct IP block to IP block connections, with multiplexers for a few bypasses. This is obviously hard to beat in terms of area and power dissipation (maximum correlation of data). In this section we look at the additional cost to provide the flexibility of a second HSEL and flexible IP block interconnection provided by the NOC.

5.1 Frequency

The original companion chip runs at 100 MHz in a CMOS 0.13 micron technology. In the extended version, the IP blocks run at 100 MHz, and the NOC at 300 MHz, in a CMOS 0.13 micron technology. The NOC can run up to 500 MHz but the raw link

bandwidth of 1.2 Gbit/sec (32 bits times 300 MHz) is sufficient for the task graphs.

5.2 Area

The area of the new companion chip is 4% larger than the original. Figure 3 shows that the routers cause 22% of the increase in area, the NI kernels 60%, and the NI shells 18%. The area numbers of the NI kernels and routers are produced by the Æthereal design flow, based on extrapolations of a number of router and NI instances (testable, lay-out with back-annotated timing). These area numbers are for a NOC running at 500 MHz; they will be an over-estimate because here the NOC runs at only 300 MHz. The routers support both guaranteed-throughput (GT) and best-effort (BE) traffic. The BE traffic class is only used for configuration, and using GT also for this traffic would reduce the router area by a factor 4 [12]. The NI kernels use most area (60%) because they implement the per-connection buffers at the edge of the network, containing a total of 2608 32-bit words. There are a total of 46 NI ports each supporting one connection (38 functional ports, and one port per NI (8 total) for programming the NOC). With two queues per port (one for each of the request and response traffic), queues are on average 28 words deep (2608 words / ((38+8) $\times$ 2) queues).

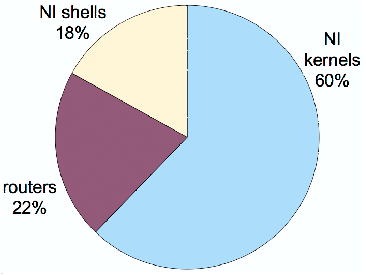

Figure 3. Relative contributions to increase in area.

5.3 Power dissipation

The Æthereal design flow produces power dissipation estimates through flit-level simulation of the NOC [8] and then computing the power dissipation of the entire NOC using the power dissipation models of the NOC components. For the most demanding task graph, the power dissipation increases by 12%. Figure 4 shows that the largest percentage (54%) is due to the NOC clock, followed by the NOC wires (18%). Wire lengths of IP block to NI of 0.8mm, and intra-NOC wires of 1.1mm were used. The high cost of clocking is caused by Æthereal's synchronous implementation, to offer performance guarantees (guaranteed throughput and latency). Note that the frequency of the NOC is constant for all task graphs, and dimensioned for the worst case (most demanding task graph). Promising recent results show that scaling the frequency and operating voltage of the NOC based on the required load per task graph can lead to average savings of 50% [17]. A 6% increase in power dissipation is acceptable for stationary systems.

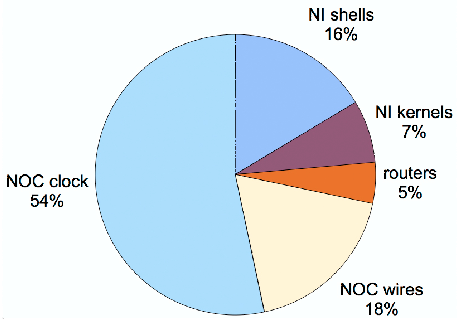

Figure 4. Relative contributions to increase in power dissipation.

5.4 Latency

Latency is not a major concern in streaming systems, such as the video-processing application. But in any case, for the most demanding task graph, the additional latency introduced by the NOC for the whole processing pipeline of the companion chip (HSEL input to HSEL output) is 126 microseconds, or just 10%. Part of the increase in latency is due to the NOC latency, but most is caused by the fact that the connections in the task graphs are not dimensioned for the *peak bandwidth* of the IP blocks, but for the *average bandwidth*. As a result, non-uniform processing speeds (in particular, above-average bursts) increase the total latency. Allocating more than average bandwidth will reduce the total latency. In computing the performance of the system as a whole, both the guaranteed communication performance and the computation performance must be taken into account therefore [2]. The latency discussed above does not include the time to configure the NOC, which is 118 microseconds, because this happens infrequently (when the task graph changes).

5.5 Simulation and verification

It requires some 10 hours to simulate one standard-definition frame on the RTL VHDL of the original companion chip and a high-level C model of the main chip. The extended version of the companion chip simulates 15% to 60% slower, depending on the task graph. This is due to the additional circuitry introduced by the NOC. Moreover, the NOC clock is a factor three faster than the original clock, leading to an increase in the number of simulation events.

The analytical performance verification tool [10], which is part of the Æthereal design flow takes only seconds to verify a task graph. The SOC performance (the timing of the data) can be verified analytically, but limited simulations are still required to verify the functional behaviour (the data values).

5.6 Design flow

The Æthereal NOC design flow comprises NOC generation, configuration, simulation, and performance verification [11]. As described in Section 4.4, all but the first step were used. The NOC was dimensioned manually, together with an IP block port to NI port mapping. Recent enhancements to the design flow automate

multi-task-graph NOC mapping and configuration [14, 16, 17], removing the need for manual mapping. The NOC topology and IP mapping are specified in XML from which an RTL VHDL implementation of the NOC, test benches, etc. as well as SystemC transaction-level models are then automatically created.

The automatic configuration (path finding, TDMA slot allocation), performance verification (which includes NI kernel buffer sizing), and simulation were all extensively used. Adding a new task graph is quick and easy: run the configuration and verification tools, and compile the embedded configuration software for the embedded processor. If a new task graph does not fit on the given NOC, the configuration and verification tools give a warning.

Without the automation provided by the design flow the new companion chip architecture would not have been defined, implemented, and verified in only 12 months by two designers who were new to both NOCs and picture-improvement SOCs.

5.7 Flexibility

The increased area, power dissipation, and latency are the price we pay for the increased flexibility and re-use potential of the companion chip. In combination with, for example, an FPGA companion chip, two new IP blocks (or sub task graphs) can replace (upgrade) any IP block in the processing pipeline on the companion chip. Alternatively new functionality can be spliced in at any point in the existing processing pipeline.

6 Conclusions

For the high-volume consumer-electronics domain, we have shown that splitting a single complex application-specific standard product (ASSP) in a smaller main chip with (multiple) companion chips is advantageous. In particular, if all the chips are NOC-based, then the NOCs on the different chips can behave transparently like a single NOC, easing the creation of a single multi-chip system.

In this context, we presented the application of the Æthereal NOC to a commercially-available high-performance SOC for TV image improvement. This SOC is a companion chip to the PNX8550 (Viper2) SOC. Retrofitting a NOC in an existing architecture, optimised for a different application-specific interconnect (wires and a by-pass) only required the creation of a new network interface (NI) shell. The existing Æthereal design flow enabled the design and implementation of a fully-functional RTL VDHL system to be verified by multi-chip system simulations, as well as analytical performance verification. The unoptimised new SOC architecture, containing a NOC, is only 4% larger in area, and has a 12% higher power dissipation than the original design. It was designed, implemented, and verified in only 12 person months.

The quantitive results indicate that changing an existing optimised architecture to use a NOC is already beneficial at acceptable cost. Architectures designed from the start with a NOC interconnect are sure to gain even more. The results indicate NOCs are a technology that can soon be used in commercial products.

References

[1] A. A. Abbo, et al. Power consumption of performance-scaled SIMD processors. In *PATMOS*, 2004.

[2] M. Bekooij, et al. Predictable embedded multiprocessor system design. In *SCOPES*, 2004.

[3] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.

[4] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *DATE*, 2005.

[5] E. Bolotin, et al. QNoC: QoS architecture and design process for network on chip. *J. of Systems Architecture*, 50(2–3), Feb. 2004.

[6] A. Danilin, M. Bennebroek, and S. Sawitzki. A novel toolset for the development of FPGA-like reconfigurable logic. In *FPL*, 2005.

[7] J. A. de Oliveira and H. van Antwerpen. The Philips Nexperia digital video platform. In G. Martin and H. Chang, editors, *Winning the SoC Revolution*. Kluwer Academic, 2003.

[8] J. Dielissen, et al. Power measurements and analysis of a network on chip. Technical Note 2005/00282, Philips Research, Apr. 2005.

[9] O. P. Gangwal, et al. Understanding video pixel processing applications for flexible implementations. In *Euromicro*, 2003.

[10] O. P. Gangwal, et al. Building predictable systems on chip: An analysis of guaranteed communication in the Æthereal network on chip. In P. van der Stok, editor, *Dynamic and Robust Streaming In And Between Connected Consumer-Electronics Devices*, volume 3 of *Philips Research Book Series*, chapter 1. Springer, 2005.

[11] K. Goossens, et al. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *DATE*, 2005.

[12] K. Goossens, J. Dielissen, and A. Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):21–31, Sept-Oct 2005.

[13] K. Goossens, et al. Interconnect and memory organization in SOCs for advanced set-top boxes and TV — Evolution, analysis, and trends. In J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, editors, *Interconnect-Centric Design for Advanced SoC and NoC*, chapter 15, pages 399–423. Kluwer, 2004.

[14] A. Hansson, et al. A unified approach to constrained mapping and routing on network-on-chip architectures. In *CODES+ISSS*, 2005.

[15] T. Marescaux, et al. Interconnection networks enable fine-grain dynamic multitasking on FPGAs. *FPL*, 2002.

[16] S. Murali, et al. A methodology for mapping multiple use-cases on to networks on chip. In *DATE*, 2006.

[17] S. Murali, et al. Mapping and configuration methods for multi-use-case networks on chips. In *ASP-DAC*, 2006.

[18] V. Nollet, et al. Centralized run-time resource management in a network-on-chip containing reconfigurable hardware tiles. In *DATE*, 2005.

[19] PCI-SIG. *PCI Express Base Specification Revision 1.0a*, Apr. 2003.

[20] R. Peset Llopis. Is there a future for differentiating ICs for high-end televisions? Presented at the International Seminar on Application-Specific Multi-Processor SoC (MPSOC), July 2004.

[21] Philips. *Nexperia PNX8550 Home Entertainment Engine*, Dec. 2003.

[22] Philips. *Nexperia PNX15xx Series Data Book*, Dec. 2004.

[23] Philips Semiconductors. *Device Transaction Level (DTL) Protocol Specification. Version 2.2*, July 2002.

[24] S. Rathnam and G. Slavenburg. Processing the new world of interactive media. *Signal Processing Magazine,*, 15(2), Mar. 1998.

[25] A. Rădulescu, et al. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 24(1):4–17, Jan. 2005.