Design of Networks-on-Chip for Real-Time Multi-Processor Systems-on-Chip

Jens Sparsø
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Email: jsp@imm.dtu.dk

***Abstract*—This paper addresses the design of networks-on-chips for use in multi-processor systems-on-chips – the hardware platforms used in embedded systems. These platforms typically have to guarantee real-time properties, and as the network is a shared resource, it has to provide service guarantees (bandwidth and/or latency) to different communication flows. The paper reviews some past work in this field and the lessons learned, and the paper discusses ongoing research conducted as part of the project "Time-predictable Multi-Core Architecture for Embedded Systems" (T-CREST), supported by the European Commissions seventh framework programme. The aim of this project is to develop a general-purpose multi-core platform for real-time systems as well as tools supporting its use (compiler, simulator, and worst-case execution time analysis tool).**

***Index Terms*—Multiprocessor interconnection networks; Real time systems; Asynchronous circuits; Time division multiplexing;**

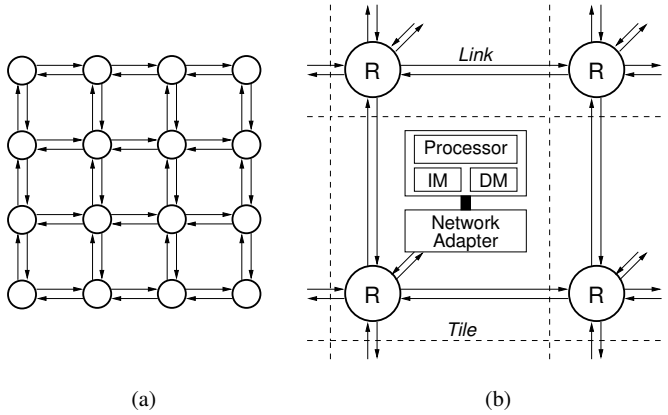

Fig. 1. An example NOC based MPSoC: (a) 2D mesh NOC topology. (b) Details of a node/tile.

I. Introduction

Over the last decade, the network-on-chip (NOC) concept has evolved from an academic research topic towards industrial take-up. Many of today's chip multi-processors (CMP) used in general purpose computing, and many of todays multi-processor system-on-chip (MPSOC) platforms used in application-specific embedded systems are built around some form of intra-chip packet-switched interconnection network. Fig. 1 shows such multi-processor platform using a 2D mesh topology NOC.

Despite the architectural convergence between CMPs and MPSOCs there are also some key differences. In general purpose computing, the processing nodes are typically homogeneous, the focus is on optimizing the typical-case performance, and the network typically supports connection-less best-effort packet-switching.

In the field of embedded systems the processing nodes are often heterogeneous, and in order to support provision of real-time requirements the network-on-chip has to provide bandwidth and/or latency guarantees. This calls for rather different NOC designs typically implementing some form of virtual circuit switching. Another observation is that much of the research in NOCs for real-time MPSoC systems targets the creation of application-specific hardware platforms. This includes application-specific NOC topologies and application-specific dimensioning of resources. The very high and growing cost of developing and fabricating large integrated circuits makes this relevant only for high volume (consumer) products. Systems that are to be manufactured in smaller quantities must therefore be based on more general-purpose platforms.

These two views: designing a custom platform targeting a specific application or using a general-purpose platform to implement a given application, may lead to somewhat different NOC-designs. The T-CREST platform, as discussed at the end of this paper, belongs to the latter category.

The paper is organized as follows. Section 2 introduces the basics of on-chip networks. Section 3 addresses how to provide real-time guarantees. Section 4 presents a number of representative NOC-designs specifically targeting real-time systems. Section 5 discusses the experience gained from this research – the baseline for our work on the NOC for the T-CREST platform. Section 6 presents the T-CREST project and the directions taken for the NOC design. Finally section 7 concludes the paper. The paper has been been written to supplement an invited talk.

II. NOC-based multi-processor systems

Fig. 1 shows an example of a NOC-based MPSoC platform. The NOC itself consists of a structure of *routers* and *links* implementing a packet-switched communication fabric, and a set of *network adapters* (NA) that bridges from the packet-switched interconnect to the memory-style read-write transaction interfaces offered to the *nodes* connected by the NOC. The network adapter maps a read or a write transaction into a packet and this typically includes generating a header

that is used to route the packet through the NOC. A packet consists of a sequence of flits that are transmitted in immediate succession. Very often the physical implementation of the entire platform use a *tile-based* approach, where tiles are simply replicated and where connections are established by abutment. A node is typically a processor with some amount of local memory (cache memories and/or explicitly managed scratchpad memories). A node can also be a slave device, for example an SDRAM-controller providing access to some larger and shared off-chip memory or an IO-controller.

The shift towards packet-switched NOC-based interconnect is due to a number of factors including those discussed below.

Natural segmentation and sharing of wires: In current and future silicon technologies it is not possible to have wires that span an entire chip. The load capacitance would be prohibitive and the signal propagation delay (latency) would be very large. Long wires have to be divided into smaller segments with repeater-buffers driving each section. An obvious next step is to add pipeline registers to the buffer stages. In this way the throughput can be improved to a point where it matches (or even exceeds) the frequency of the clock signals used in the processors nodes. Finally, by adding multiplexors to the input of the pipeline registers, a X-Y switched pipelined interconnect emerges – i.e. a simple network.

GALS-style timing organization: In order to be practically feasible, a large chip has to be designed as a globally-asynchronous and locally synchronous system [1]. The tile-based organization of a multi-processor platform offers spatially confined components with clean interfaces where it is natural to introduce clock domain crossings: (i) processor to network adapter interface, (ii) network adapter to router interface , and (iii) router to router interface. Furthermore it seems natural to seek implementations of the routers and links which can tolerate a fixed clock skew (mesochronous), a drifting clock skew (plesiochronous), or perhaps even more natural, asynchronous implementations possibly using delay-insensitive signaling on the links.

Bandwidth scales with the number of nodes: When the number of nodes grows, the number of links and routers grows as well and this provides more bandwidth. In regular two-dimensional topologies, like mesh or torus, the number of links grows linearly with the number of nodes (N) and the bisection bandwidth grows as the square-root of N. Finally the packet switching means that the wires implementing a link can be shared by several communication flows.

Modular design: The NOC offers simple plug and play composability. Assuming standardized interfaces (processor-NA, NA-router, and router-router) large systems with any number of processor nodes and any network topology can are built by simply connecting the necessary components. This is the hardware view. There is also a software side to this, which is perhaps less trivial.

III. Real-time NOCs – basic principles

The fact that a NOC is a shared communication medium comprising multiple independently-arbitrated resources (routers or links) may severely complicate timing analysis. The seemingly simple question: ”What is the latency that the NOC adds to a read or write transaction towards a memory in a remote node?” can be very difficult to answer, and this makes it hard to compute the worst-case execution time (WCET) of a program executing on a processor node and thus guaranteeing real-time behavior. Many NOCs are designed to support best-effort traffic only and several of these (e.g. QNOC [2] and ANOC [3]) offer multiple priority-levels for packets, but hard real-time guarantees are not a property of the NOC itself.

A NOC for a real-time platform must allow guarantees on bandwidth and/or latency to be made, and this calls for some form of end-to-end connection. There are essentially two ways of achieving this: non-blocking routers with rate control (e.g. Mango [4], [5]), and circuit switching (e.g. SoCBUS [6] and Wolkotte [7]), possibly with time division multiplexing (TDM), (e.g. Æthereal [8], [9] and Nostrum [10]). In the following section we present and discuss each of these NOCs. They differ in whether or not they are work conserving and to what extent hardware resources are reused for different virtual circuits.

IV. real-time NOCs – case studies.

MANGO was designed with the following goals in mind: (i) it should be implemented using asynchronous circuits in order to support a GALS-style platform, (ii) it should be simple in order to facilitate high operation speed and low hardware overhead, (iii) it should support both best-effort (BE) and guaranteed service (GS) traffic, (iv) it should be work-conserving in order to make efficient use of network resources, and finally (v) it should provide bounds on latency and bandwidth which are decoupled (or at least not inversely dependent on each other). The latter two goals are in contrast to TDM-based NOC’s like NOSTRUM and Æthereal. GS and BE traffic is handled by separate resources in the router. The GS part uses rate control and a non-blocking router. The latter requires a separate physical buffer for each virtual circuit (VC) sharing an outgoing link. These VC-buffers are in the output ports, i.e. after the crossbar switch and before the link arbitration module. To regulate the flow of flits, MANGO uses an elaborate arbitration mechanism for each router output port supplemented by a credit-based flow control mechanism among the VC-buffers buffers in neighboring routers, along the path that implements the end-to-end virtual circuit [5]. In this way GS-traffic is routed through a sequence of VC-buffers and the path is set up at initialization time by programming the crossbar switches in the GS-routers. Packets are therefore header-less. MANGO is work conserving, the links are shared but the VC-buffers in the routers are not.

SoCBUS uses pure circuit-switching. No resources, i.e. wires, and router buffers are shared between connections. This lowers utilization and increases cost. However, once a connection has been established, real-time guarantees are trivially achieved. SoCBUS uses a ”dial up” scheme where special packets are used to establish and tear down connections. It

may happen that a requested connection cannot be established because some resource along the path is already reserved by another connection. The fact that this is possible compromises the real-time properties. As all resources for a given end-to-end circuit are private it follows that the design is work conserving.

The NOC described by Wolcotte et al. in [7] is also a purely circuit-switched design – but more in the sense of a reconfigurable coarse-grained FPGA. The design uses a mesh-type structure of routers and links. Each link is divided into a number of lanes, and the router (a statically configured crossbar) may be set to switch theses lanes individually. Depending on bandwidth requirements a connection may use one or more lanes. The approach may be advantageous in streaming-type application. As is the case for SoCBUS no resources, i.e. wires, and router buffers are shared between connections.

Æthereal uses time division multiplexing (TDM). In each time-slot a router simply forwards data/packets from its input ports to its output ports. The network adapters inject packets into the packet switched structure of routers and links according to a pre-determined schedule, which avoids the need for arbitration, flow control, and buffering – the routers and links form a simple switched pipelined circuit. The network adapters implement credit-based flow control for end-to-end virtual circuits. The original Æthereal router design [9] supports both GS and BE traffic and the slot tables, controlling the switching of the GS-traffic through the routers, are implemented inside the routers. BE-traffic uses slots that are not reserved and slots that are reserved but not used. A light version of Æthereal, called aelite, supports GS traffic only, and it uses source routing where packet headers carry the routing information [11]. This eliminates the slot-tables from the routes, and the result is a very simple router design. In the latest version of aelite, called dAElite [12], the routing tables are again placed in the routers in order to support multi-cast routing. All versions of Æthereal are non-work conserving, but all resources that a packet traverses are shared.

NOSTRUM [10] is another example of a pioneering NOC based on time division multiplexing. It uses a concept of looping containers to support GS traffic. A container can contain a packet or it can be empty. Bandwidth for a given virtual circuit is reserved by ensuring that enough containers are looping from source to destination. The fact that containers loop may be used to support acknowledged writes and transmission of read responses.

V. Real-time NOCs – Discussion

In retrospect, and in this authors judgment, MANGO largely met the five goals listed. The only real downside is the hardware cost, which is dominated by the many VC buffers and the large crossbar switch. A typical router for a 2D-mesh NOC has 5 ports with 8 VC's per port (7 for GS and 1 for BE). This results in a 5×40 crossbar switch, and 40 VC buffers each 2-4 flits deep. Assuming 32-bit links the total buffer capacity of a router is in the order of 2500-5000 bits. According to figures presented in [13] this accounts for 19.5% of the router area. The corresponding crossbar switch accounts for 34.7 % of the area. The entire router has an area of 0.277 mm^2 in a 0.13 μm CMOS technology and its speed range from 400 Mflits/s (assuming worst-case parameters and delay-insensitive signaling on the links) to 1,000 Mflits/s (assuming typical-case parameters and bundled-data signaling on the links). An alternative design with 5 VCs per port has an area of 0.188 mm^2 in a 0.13 μm CMOS technology. These figures are comparable with those reported for the original Æthereal NOC [9].

A comparable 5 ported aelite router, implemented as a 3-stage pipeline and using 32 bit links has a total of 480 bits of pipeline registers, and as there are no virtual circuit buffers the crossbar switch is the simplest possible. Given that the control is also simpler – because there is no flow control – it is seems safe to estimate a 10:1 area difference between MANGO and aelite. Actual numbers for the aelite design is reported in [11] for an implementation in 90 nm CMOS technology, but it is difficult to scale and compare across technologies. A TDM-based NOC requires some form of common time reference to control the time-slotting, and this favors a simple clocked synchronous implementation. As mentioned before, the NOC is a chip-wide structure, and synchronous operation is generally not possible. Mesochronous or fully asynchronous solutions are needed. A mesochronous implementation of aelite is reported in [11]. It adds small FIFO-buffers in the links in order to compensate for the phase difference between the clocks in the routers connected by the link. A mesochronous design that can tolerate clock skew of up to half a clock period is obtained by adding 4-word deep FIFO's to the links, and this results in a 50% area increase. This reduces the area advantage to 7:1 but this is still significant.

The above discussion suggests that perhaps a simpler and faster design that provides plenty of raw bandwidth is a better choice than a more elaborate design. TDM is not work conserving, and many slots on a given channel may not be used, resulting in a poorer bandwidth utilization. Given the very large speed-performance advantage, and given that the design of a real-time system is driven by worst-case concerns, this may not matter much.

While the design of routers have received a lot of attention in the literature, it is interesting to observe that there is surprisingly little data published on the hardware complexity of the network adapters. Looking at Fig. 1(b), and keeping in mind that the processing is done in the processor node, the router and the network adapter in a tile represents a "communication overhead". And when using a simple TDM-based router (like aelite), the network adapter may well stand out – its hardware complexity may even be comparable to that of a simple processor. In this authors opinion, it is worthwhile taking a closer look at this issue. For the original Æthereal NOC the network adapter is reported to have an area of 0.11 mm^2 in a 0.13 μm CMOS technology [14]. According to [15] an area-optimized 32-bit MIPS M4K processor has an area of 0.185 mm^2 in a 0.13 μm CMOS technology.

Finally a few comments on the nature of the design challenges faced when designing a NOC. The packet-switched interconnect consisting of routers and links simply transports packets from source to destination. Here, the challenges are mostly hardware design issues like area, speed and power consumption of the routers and links, as well as the use of mesochronous, plesiochronous or asynchronous circuit design solutions. The design of the network adapters on the other hand is heavily influenced by system-level issues like the programming model and the memory hierarchy. The memory model is typically distributed non-coherent shared memory. In in order to hide the latency associated with accessing memory in a remote node, DMA-controllers are often used to transfer blocks of data. This functionality may be provided by the network adapter or it may be provided by the processor node – the boundary between the two is not 100% obvious.

The issues discussed here form the baseline for our current NOC design work in the T-CREST project.

VI. The T-CREST project

The T-CREST project aims at developing a multi-processor platform for real-time systems where all components (processors, interconnection network, compiler etc.) are designed with a focus on time-predictability, e.g., optimized to minimize the WCET and to allow analysis tools to provide tight bounds on the WCET.

The hardware platform will be similar to the one shown in Fig. 1. We envision a prototype with up to 64 processor nodes. Each processor node will contain local instruction and data caches and local scratchpad memories. The platform will also include a memory controller providing access to an off-chip SDRAM memory. The scratchpad memories in the nodes and the off-chip SDRAM memory is mapped into a single shared address space. Each processor node will have a number of DMA controllers which are capable of performing block read and write transaction targeting the external SDRAM memory as well as scratchpad memories in remote nodes. Effective use of the DMA controllers for overlapping of processing and data transfer is crucial for hiding the considerable latency of the NOC and the the shared memory controller.

The T-CREST project includes work packages addressing: the processor, the NOC, the compiler, the memory controller, and the WCET analysis tool (adaption of an existing WCET tool), and evaluation of the platform using two industrial applications.

The processor is a statically scheduled, dual-issue RISC processor that is optimized for real-time systems [16]. The execution times of the individual instructions are known and are provided as part of the instruction set architecture (ISA). Instructions are predicated in order to support the generation of code with known and fixed execution time. To make the memory hierarchy equally time-predictable as well it is envisioned that the processor will use several specialized caches: an instruction cache which operates on whole functions/methods [17] and a set of specialized data caches (a stack cache, a cache for for static data, constants, and a small, fully associative cache for heap access). Cache misses, as well as explicitly initiated DMA transfers, access the off-chip SDRAM through a time-predictable memory controller.

The development of the T-CREST NOC will be based on the experiences from work on the MANGO and AEthereal NOC's. As mentioned in the introduction, multi-processor platforms for embedded systems are typically optimized for a given application or application domain. In the T-CREST project our aim is to develop a general-purpose platform – either a platform that can be configured to optimize the performance of the system or a platform that can be used as is, without any configuration. Our aim is to develop an asynchronous version of a simple TDM-based NOC. The NOC is used to transfer different types of data: message passing between cores, cache fills from main memory, DMA-driven communication between scratch-pad memories and the off-chip memory, and synchronization operations such as compare-and-swap. In some architectures a single NoC serves all those different types of data. However, the requirements of these various types of data with respect to packet size, address ranges, and flow control are different. Therefore, T-CREST will evaluate if several different NOCs, each optimized for some type(s) of traffic, represents a better choice. As an example we mention that the scratchpad to scratchpad message passing in the T-CREST project requires an "all-to-all" network (which can be TDM-based, without buffering and flow-control all the way from the source to the destination), whereas the traffic towards the memory controller managing the off-chip SDRAM requires an "all-to-one" network. Initial ideas for a synchronous general-purpose all-to-all TDM-based NOC design have been explored in [18].

VII. Conclusion

This paper reviewed some fundamentals issues in the design of networks on chips, presented and discussed a number of NOCs specifically developed for use in real-time systems and presented the NOC-design considerations in the T-CREST project. The paper argued for simple and fast solutions.

Acknowledgement

This work was partially funded under the European Union's 7th Framework Programme under grant agreement no. 288008: Time-predictable Multi-Core Architecture for Embedded Systems (T-CREST).

We would like to thank the T-CREST project members for the interesting and inspiring discussions on time-predictable architectures during the project meetings.

References

[1] The International Technology Roadmap for Semiconductors, "ITRS 2009 Edition - Design," http://www.itrs.net/, 2009.

[2] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, Mar. 2005, pp. 44 – 53.

[3] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous noc architecture providing low latency service and its multi-level design framework," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC).*, Mar. 2005, pp. 54 – 63.

[4] T. Bjerregaard and J. Sparsø, "A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip," in *Proc. Design Automation and Test in Europe (DATE)*. IEEE Computer Society Press, 2005, pp. 1226–1231.

[5] ——, "A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-chip," in *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society Press, 2005, pp. 34–43, (Best paper award).

[6] D. Wiklund and D. Liu, "SoCBUS: Switched network on chip for hard real time embedded systems," in *Proc. Int'l Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 2003, p. 78a.

[7] P. T. Wolkotte, G. J. Smit, G. K. Rauwerda, and L. T. Smit, "An energy-efficient reconfigurable circuit switched network-on-chip," in *Proc. Int'l Parallel and Distributed Processing Symposium (IPDPS)*, April 2005, p. 8 pp.

[8] K. Goossens and A. Hansson, "The aethereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proc. ACM/IEEE Design Automation Conference (DAC), 2010*, Jun. 2010, pp. 306 –311.

[9] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sept-Oct 2005.

[10] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The nostrum backbone-a communication protocol stack for networks on chip," in *Proc. International Conference on VLSI Design*, 2004, pp. 693 – 696.

[11] A. Hansson, M. Subburaman, and K. Goossens, "aelite: a flit-synchronous network on chip with composable and predictable services," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2009, pp. 250–255.

[12] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens, "A TDM NoC supporting QoS, multicast, and fast connection set-up," in *Proc. Design, Automation and Test in Europe Conference (DATE)*, 2012, pp. ??–??

[13] T. Bjerregaard and J. Sparsø, "Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip," *IEE Proceedings: Computing and Digital Techniques*, vol. 153, no. 4, pp. 217–229, 2006.

[14] A. Rădulescu, J. Dielissen, K. Goossens, E. Rijpkema, and W. Paul, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration," in *Proc. Design, Automation and Test in Europe (DATE)*. IEEE Computer Society Press, 2004, pp. 878–883.

[15] Berkeley Design Technology, Inc., "MIPS Technologies MIPS32 M4K Synthesizable Processor Core," Tech. Rep., 2007. [Online]. Available: http://www.mips.com/media/files/m4k/FINAL_mips_m4k.pdf

[16] M. Schoeberl, P. Schleuniger, W. Puffitsch, F. Brandner, C. W. Probst, S. Karlsson, and T. Thorn, "Towards a time-predictable dual-issue microprocessor: The Patmos approach," in *First Workshop on Bringing Theory to Practice: Predictability and Performance in Embedded Systems (PPES 2011)*, Grenoble, France, March 2011, pp. 11–20.

[17] M. Schoeberl, "A time predictable instruction cache for a Java processor," in *On the Move to Meaningful Internet Systems 2004: Workshop on Java Technologies for Real-Time and Embedded Systems (JTRES 2004)*, ser. LNCS, vol. 3292. Agia Napa, Cyprus: Springer, Oct. 2004, pp. 371–382.

[18] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki, "A Statically Scheduled Time-Division-Multiplexed Network-on-Chip for Real-Time Systems," in *Proc. IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE Computer Society Press, May 2012, pp. ??–??, (Accepted).