An Area-efficient Network Interface for a TDM-based Network-on-Chip

Jens Sparsø, Evangelia Kasapaki, Martin Schoeberl
Department of Informatics and Mathematical Modelling
Technical University of Denmark
Email: jsp@imm.dtu.dk, evka@imm.dtu.dk, masca@imm.dtu.dk

***Abstract*—Network interfaces (NIs) are used in multi-core systems where they connect processors, memories, and other IP-cores to a packet switched Network-on-Chip (NOC). The functionality of a NI is to bridge between the read/write transaction interfaces used by the cores and the packet-streaming interface used by the routers and links in the NOC. The paper addresses the design of a NI for a NOC that uses time division multiplexing (TDM).**

By keeping the essence of TDM in mind, we have developed a new area-efficient NI micro-architecture. The new design completely eliminates the need for FIFO buffers and credit based flow control – resources which are reported to account for 50-85 % of the area in existing NI designs. The paper discusses the design considerations, presents the new NI micro-architecture, and reports area figures for a range of implementations.

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

I. Introduction

Over the last decade, the network-on-chip (NOC) concept has evolved from an academic research topic towards industrial take-up and most of today's multi-core platforms uses some form of packet-switched on-chip interconnect. Fig. 1 shows an example of such a platform using a 2D-mesh topology. Typical cores are: (i) processors with some amount of local memory, (ii) memory controllers for external shared memory, (iii) dedicated hardware accelerators, and (iv) IO devices.

As illustrated in Fig. 1(b), the network interface (NI) bridges between the address-space read-write transaction interface used by the cores and the packet streaming interface used by the routers in the network. While there are standards for the former (AMBA AXI [1], OCP [2], etc.), the packet interface is specific for the specific NOC. As the NI encapsulates these NOC-specific details, it follows that NIs, or at least the back-ends of NIs, may be as different as the routers that are used in the NOC. There are surprisingly few papers addressing the design of NI's, and often the area measures reported can be difficult to compare.

The work presented in this paper is part of the larger project T-CREST [3] aiming at developing a time-predictable, NOC-based multi-core platform for hard real-time systems. This context implies an emphasis on worst-case execution time, which must be analyzable/predictable and with a tight bound in order to avoid over-engineering. The entire design, including the processor cores, the NOC, the memory architecture, and the compiler is being designed with the aim of minimizing the worst-case rather than average-case execution time.

978-3-9815370-0-0/DATE13/ © 2013 EDAA

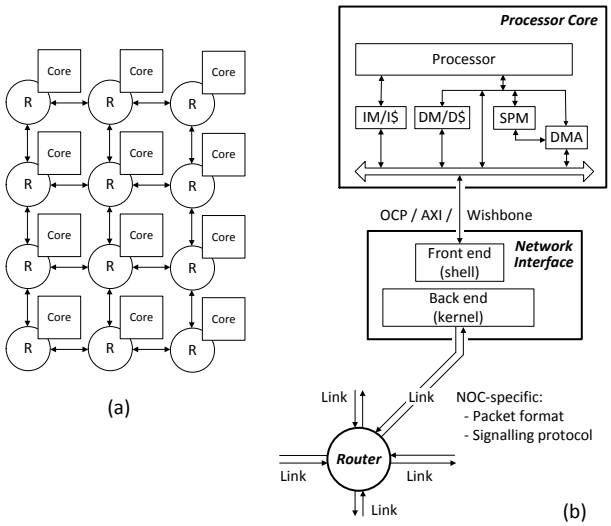

Fig. 1. An example NOC based MPSoC: (a) 2D mesh NOC topology. (b) Details of a processor core consisting of a processor with local memories (caches and/or scratchpad memories), one or more DMA controllers, and a network adaptor.

This focus on time-predictability, in combination with an aim of keeping the hardware cost low, has caused us to adopt ideas from the time-division-multiplexing (TDM) based NOC designs (e.g., aelite [4]). The TDM scheme supports time predictability in a straightforward way, and the routers and links are extremely simple and efficient.

To an application programmer the inter-processor NOC offers end-to-end circuits between any two cores. In a typical application the "connectivity" between processor cores is sparse and the TDM schedule period correspondingly short, but as demonstrated in [5], even a fully connected topology can be supported by a TDM schedule with a modest period.

Our NOC uses source routing and the NIs inject packets into the packet switched structure of routers and links according to a pre-determined periodic TDM-schedule, which avoids the need for dynamic arbitration, buffering, and flow control. In this way the routers and links form a simple, switched, and pipelined circuit. From a hardware point of view it does not get much simpler.

The contribution of this paper is an area-efficient NI micro-architecture. It has evolved from reconsidering a number of

issues including the interface between a processor core and the NI, the clock-domain crossings, and the way DMA controllers are used to implement end-to-end transfer of data between processor cores. The key idea underlying the new NI micro-architecture is to apply TDM-based scheduling from end-to-end. This means that reading from a source memory, traversing the source NI, traversing the packet switched NOC (routers and links), traversing the destination NI, and writing into the destination memory is based on a global, static TDM schedule. With this end-to-end TDM-schedule we benefit from the very essence of TDM-based multiplexing that avoids buffering, flow-control, and dynamic arbitration. A key element of the new NI is that DMA controllers, which are normally part of the processor cores, are moved into the NI, and are implemented with an efficient table structure.

The paper is organized as follows. Sec. II presents related work. Sec. III presents our new NI architecture and the underlying observations and ideas. Sec. IV presents results on area and speed for a prototype implemented in FPGA technology and in a 90 nm CMOS standard-cell library. Finally, Sec. V concludes the paper.

II. Related Work

The fact that a NOC is a shared communication medium comprising independently arbitrated resources (routers or links) severely complicates timing analysis. In order to give guarantees on bandwidth and/or latency some form of end-to-end connections are needed. Solutions to this include non-blocking routers with rate control (e.g., Mango [6]), and circuit switching (e.g., SoCBUS [7]), possibly with time division multiplexing (TDM), (e.g., Æthereal [8], [4]).

Details of a NI for the Spidergon NOC developed by ST Microelectronics are reported in [9], and the paper compares against a number of other NI designs. The area measures reported in [9] for typical size NIs range from 7 to 50 kgates. In [10] the area of a typical NI instance for the original Æthereal NOC, which supports both GS and BE traffic, is reported to be 0.25 mm^2 in a 0.130 μm CMOS technology. In [9] this is quoted as 21 kgates.

A specific aelite NOC instance has been designed for a TV-set platform, and a discussion of its implementation cost is provided in [11, Sect. 8.1]. The NOC has 53 physical ports and supports 45 connections (bi-directional channels) among the 11 cores that it connects. The NOC comprises 6 routers, 11 NIs and 54 shells. In a 90 nm CMOS technology the entire NOC occupies a cell area of 5.5 mm^2. Buffers in the NIs related to the channel endpoints totals 24 KB and account for 4.7 mm^2 (85 %). This figure assumes a flip-flop based buffer design. Based on the figures reported in [11] we calculate the average area of one of the 11 NIs to be 0.49 mm^2 when using flip-flop based FIFOs, 0.22 mm^2 when using SRAM based FIFOs, and 0.13 mm^2 when using custom FIFOs.

An interesting perspective on the above area measures is to compare against the size of a synthesizable 32-bit processor. In a 90 nm technology the size of a mips32-m14kc processor is 0.2-0.5 mm^2 [12]. This includes the CPU itself, the system co-processor, a memory management unit, a translation look-aside buffer, a multiply-divide unit, 8 KB I-cache and 8 KB D-cache. These figures span the same range as the above mentioned NI designs. It is interesting to observe that the existing literature seem not to question the size of the published NI-designs.

III. The TDM-based Network Interface

Fig. 1(b) shows a processor with some private memory (caches and explicitly managed scratchpad memories (SPM)) and a DMA controller. In our T-CREST platform, as well as in the CoMPSoC platform [13], the DMA controllers are intended to implement background DMA-driven block transfers from the local SPM and into the SPM of a remote processor. This implements message passing. Our new NI micro-architecture, Fig. 2, pulls the DMA controllers into the NI and integrates them with the TDM scheduling mechanism.

A. Baseline Observations

The very essence of time-division-multiplexing is that it avoids buffering, flow-control, and dynamic arbitration. Ideally this implies that it should be possible to transfer data all the way from the SPM of one processor core into the SPM of another processor core without any buffering, flow control, and dynamic arbitration.

The routers in the aelite NOC enjoy this simplicity, but the NIs in aelite does not, and we are not aware of any NI design that actually enjoys this simplicity. The essence of the problem is the following: Conceptually data is communicated over an end-to-end virtual circuit. In reality this involves a number of hops, not just from router to router, but also inside the NIs due to the layered implementation of these. Timing uncertainties between a source and a sink involved in a hop anywhere along the connection may compromise the inherent simplicity of the TDM approach, and result in a need to introduce buffering and flow control. Our new NI design avoids these problems.

An application may want to concurrently send messages to several other processors. In a TDM-based NOC all communication channels are assigned some slots in the TDM schedule and DMA transfers have to be interleaved correspondingly. This calls for one DMA controller per outgoing channel.

On the other hand, the NI can only inject one packet at a time into the NOC and consequently only one DMA controller can be active at a time. This allows time-slicing a single DMA and sharing the resources for a number of *logical* DMAs. This enables a DMA controller design using a single memory structure that stores the address pointers, the word counts, and the status and control registers for all the DMA controllers.

B. Micro-Architecture of the TDM-based NI

Our key ideas are to use the already dual-ported communication memories (SPMs) for clock domain crossing (as in [14]), and to move the DMA controllers from the processor cores into the NIs. This avoids the need for buffering and flow control, it allows the DMA controllers to directly deliver the payload data to outgoing packets in accordance with the TDM

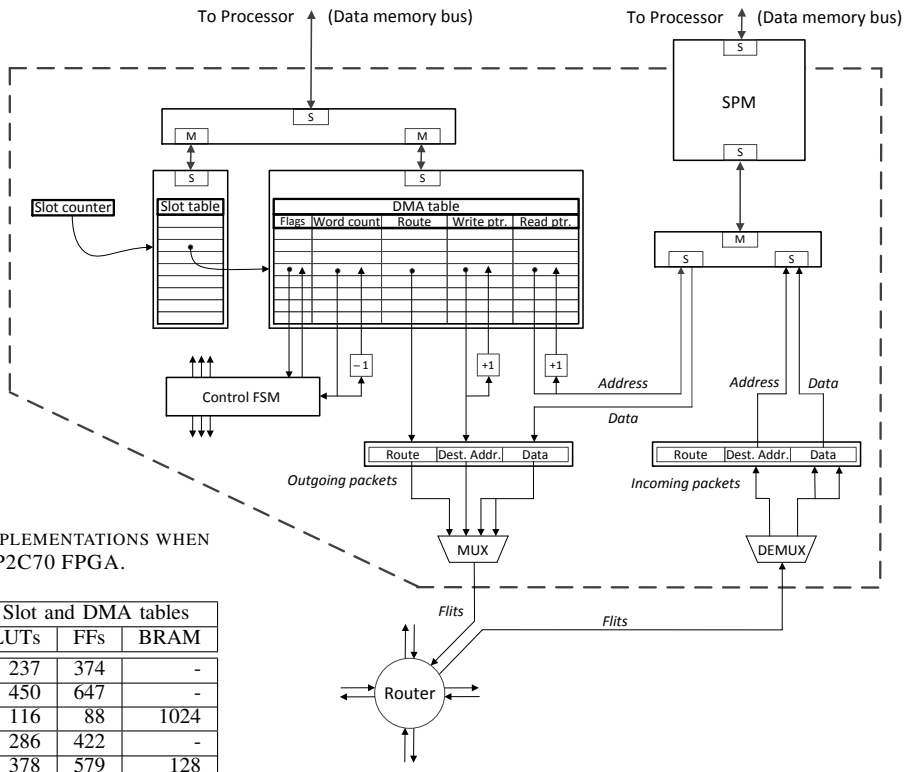

Fig. 2. Block diagram showing the micro-architecture of the new network interface. The design merges the DMA-controllers and the TDM scheduling such that end-to-end data transfer is controlled by the same global TDM-schedule.

TABLE I
AREA FIGURES FOR A SELECTION OF NI IMPLEMENTATIONS WHEN SYNTHESIZED FOR AN ALTERA EP2C70 FPGA.

NI design size		NI Logic		Slot and DMA tables		
Slot Period	DMAs	LUTs	FFs	LUTs	FFs	BRAM
16	4	326	162	237	374	-
	8	337	162	450	647	-
	16	341	162	116	88	1024
32	4	326	163	286	422	-
	8	339	163	378	579	128
	32	346	163	34	3	2240
64	4	328	164	175	323	192
	8	340	164	378	579	256
	64	351	164	34	3	4544

schedule, and it opens for a very interesting and efficient table-based implementation of the DMA controllers. This novel micro-architecture, shown in Fig. 2, is the key contribution of the paper.

The key elements of the micro-architecture are the *slot counter*, the *slot table*, and the *DMA table*. The slot counter is reset and incremented in all NIs using the same (mesochronous) clock and the slot counter defines the slots in the TDM schedule period. A slot corresponds to the time it takes to transmit a packet into the NOC, and consequently the number of clock cycles in a slot is identical to the number of flits in a packet.

The slot counter indexes a *slot table* whose entries consist of a valid bit and an index into the DMA table. The valid bit indicates whether or not a packet is to be sent in the corresponding time slot. The interfaces denoted “M” and “S” are master and slave ports supporting point-to-point read/write transactions. An entry in the DMA table holds all the registers that are found in a normal DMA controller: some control bits (start and done), a read pointer, a write pointer, and a word count. In addition to this, an entry also holds the route to be used when a packet is sent across the NOC. The reason to split between a slot table and a DMA table is to be able to assign more than one slot in a TDM schedule period to a connection and reserve different amounts of bandwidth for different end-to-end circuits.

A variation of the architecture involves storing the route information in the slot table. This would allow a connection that has been allocated several slots in the TDM period to use different routes in the different slots. Another variation merges the slot table and the DMA table into a single table. This may allow a more efficient hardware implementation, and it is possible if no connection is assigned more than one slot within the TDM period.

Finally we mention that the aelite-based CoMPSoC platform uses one communication memory per connection, while our architecture needs only a single communication memory shared by all connections in and out of the processor core. This is due to the interleaved access resulting from the TDM scheme. Consequently, the new NI micro-architecture may lead to substantial area reductions on the processor cores as well.

C. Support of a GALS Organization

Our aim is a GALS-style design allowing independently clocked processor cores. For the transfer of data between the processor and the NI the dual ported SPM provides clock domain crossing for free. Access to the DMA table crosses the clock domain boundary and a few cycles latency is added to read transactions. As setting up DMA transfers is likely to be rare events, compared to DMA initiated read or write transactions, this is a small overhead.

The TDM scheme requires that the NIs emit and absorb flits (and/or packets) at the same rate and synchronous with respect to the TDM schedule. This calls for a mesochronous clocking of the NIs. From this follows that the routers must be mesochronous as well. An entirely asynchronous implementation of the routers is also possible – the mesochronous NIs will input and output flit-tokens at the same rate, and the speed of the asynchronous routers and links has to exceed this rate for safe operation. Such a design is our final goal, and this is the reason we decided on using source routing, because it reduces the routers to simple pipelined controlled switches.

IV. Implementation and Results

To evaluate our NI design we have developed a parameterized VHDL description of a NOC using the new NI micro-architecture and a 3-stage pipelined aelite-style router [11]. The design is globally synchronous and we have not (yet) implemented any clock domain crossings. A packet consists of a header flit and two data flits each holding one 32-bit word. The header consists of the route and the local address in the target SPM and they are directly provided by the DMA table. The choice of sending two 32-bit words in one packet represents a compromise between bandwidth utilization and a desire to have a short TDM slot and thereby a short TDM schedule period.

The processor has access to the SPM, to the slot table (for initialization purposes), and to the DMA table. We envision that the system will be booted using a separate processor-memory network, and that each processor will initialize its NI before normal operation starts. Our simulation setup initializes the NIs in this way.

The NI design has been synthesized for an Altera Cyclone II EP3C70 chip. Table I shows the area figures for a selection of NI implementations ranging from a slot period of 16 and 4 DMAs up to a slot period of 64 and 64 DMAs. The table lists the amount of hardware resources used: the number of (4-input) LUTs, the number of flip-flops, and in these cases where Block RAM has been synthesized, the number of bits used by the NI design. The table shows separate figures for the NI logic and for the slot and DMA tables. As seen the size of the NI logic is constant across the different size NI instances.

In order to compare against the specific aelite NOC discussed in section II we have synthesized the NI with a slot period of 32 and 8 DMAs per NI to a 90 nm CMOS process. Using standard cells and flip-flop-based tables the total area of a NI is 0.024 mm^2, and the breakdown is NI logic 0.0063 mm^2, slot table 0.0036 mm^2 and DMA table 0.014 mm^2.

V. Conclusion

This paper presented a novel and area efficient network interface for a TDM-based NOC. The key idea is to apply a single global TDM schedule for the whole communication path from reading local memory, traversing the NI, traversing the NOC (packet switched routers and links), the target NI, and writing into the target local memory.

As a result, the new design completely eliminates the need for FIFO buffers and credit-based flow control, resources which are reported to account for 50-85 % of the area in existing network interface designs, and the new NI design is at least 2-3 times smaller than previously published NIs for real-time NOCs. This is achieved by moving the DMA controllers, which are normally implemented as part of the processor cores, into the NI and by operating the DMAs in synchrony with the TDM scheduling in the NI. The NI is intended to be used with simple TDM-based routers, resulting in a very small and efficient NOC design.

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] ARM Ltd., "AMBA Advanced eXtensible Interface (AXI) Protocol," 2011. [Online]. Available: http://www.arm.com

[2] Open Core Protocol (OCP) Specification, Release 3.0, 2009. [Online]. Available: http://www.ocpip.org

[3] Time-predictable Multi-Core Architecture for Embedded Systems (T-CREST). [Online]. Available: www.t-crest.org

[4] 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.

[5] M. Schoeberl, F. Brandner, J. Sparsø, and E. Kasapaki, "A statically scheduled time-division-multiplexed network-on-chip for real-time systems," in *Proceedings of the 6th International Symposium on Networks-on-Chip (NOCS)*. Lyngby, Denmark: IEEE, May 2012, pp. 152–160.

[6] 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.

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

[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] S. Saponara, L. Fanucci, and M. Coppola, "Design and coverage-driven verification of a novel network-interface IP macrocell for network-on-chip interconnects," *Microprocessors and Microsystems*, vol. 35, no. 6, pp. 579–592, 2011.

[10] Radulescu, Dielissen, Pestana, Gangwal, Rijpkema, Wielage, and Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, 2005.

[11] A. Hansson and K. Goossens, *On-chip interconnect with aelite / Composable and predictable systems*. Springer, 2011, embedded systems.

[12] MIPS Technologies Inc., "MIPS32 M14Kc Core." [Online]. Available: www.mips.com/products/processor-cores/classic/mips32-m14kc/

[13] A. Hansson, K. Goossens, M. Bekooij, and J. Huisken, "CoMPSoC: A Template for Composable and Predictable Multi-Processor System on Chips," *Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, 2009.

[14] M. Schoeberl, "A time-triggered network-on-chip," in *International Conference on Field-Programmable Logic and its Applications (FPL 2007)*. Amsterdam, Netherlands: IEEE, August 2007, pp. 377–382.