Project Number 288008

D 3.2 Simulation Model of the Self-timed NoC

Version 1.0
18 September 2012
Final

Public Distribution

Technical University of Denmark

Project Partners: AbsInt Angewandte Informatik, Eindhoven University of Technology, GMVIS Skysoft, Intecs, Technical University of Denmark, The Open Group, University of York, Vienna University of Technology

Project Partner Contact Information

AbsInt Angewandte Informatik Christian Ferdinand Science Park 1 66123 Saarbrücken, Germany Tel: +49 681 383600 Fax: +49 681 3836020 E-mail: ferdinand@absint.com	**Eindhoven University of Technology** Kees Goossens Potentiaal PT 9.34 Den Dolech 2 5612 AZ Eindhoven, The Netherlands E-mail: k.g.w.goossens@tue.nl
GMVIS Skysoft João Baptista Av. D. Joao II, Torre Fernao Magalhaes, 7 1998-025 Lisbon, Portugal Tel: +351 21 382 9366 E-mail: joao.baptista@gmv.com	**Intecs** Silvia Mazzini Via Forti trav. A5 Ospedaletto 56121 Pisa, Italy Tel: +39 050 965 7513 E-mail: silvia.mazzini@intecs.it
Technical University of Denmark Martin Schoeberl Richard Petersens Plads 2800 Lyngby, Denmark Tel: +45 45 25 37 43 Fax: +45 45 93 00 74 E-mail: masca@imm.dtu.dk	**The Open Group** Scott Hansen Avenue du Parc de Woluwe 56 1160 Brussels, Belgium Tel: +32 2 675 1136 Fax: +32 2 675 7721 E-mail: s.hansen@opengroup.org
University of York Neil Audsley Deramore Lane York YO10 5GH, United Kingdom Tel: +44 1904 325 500 E-mail: Neil.Audsley@cs.york.ac.uk	**Vienna University of Technology** Peter Puschner Treitlstrasse 3 1040 Vienna, Austria Tel: +43 1 58801 18227 Fax: +43 1 58801 918227 E-mail: peter@vmars.tuwien.ac.at

Contents

Document Control

Version	Status	Date
0.1	First draft	1 August 2012
1.0	Final draft	18 September 2012

Executive Summary

This document describes the deliverable *D 3.2 Simulation Model of the Self-timed NoC* of work package 3 of the T-CREST project, due 12 months after project start as stated in the Description of Work.

The document presents the architecture of a new area-efficient network interface and its implementation in the form of a register transfer level VHDL model. By using this model in conjunction with an RTL model of a router it is possible to create structural simulation models of the entire T-CREST NOC.

The source code is provided through the t-crest repository via the `git` source code management tool, and this document provides information on how to build and run the simulation model. The NOC model assumes a 2D bi-torus topology and is parametrized in the number of processor nodes, the size of TDM-slot tables, the number of DMAs in the different NIs etc. The functionality has been verified by running a number of tests that are available as part of the source code.

1 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 use some form of packet-switched on-chip interconnect. This is the case for general purpose chip multi-processors (CMP) as well as for multi-processor systems-on-chip platforms (MPSoC) that are used to implement application-specific embedded systems. The aim of the T-CREST project is to develop a prototype of a multi-processor platform-chip that supports implementation of hard real-time applications.

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 another core?" can be very difficult to answer. In order to give guarantees on bandwidth and/or latency some form of end-to-end connections are needed. As discussed in [5] solutions to this include non-blocking routers with rate control (e.g. Mango [2, 3]), and circuit switching (e.g. SoCBUS [15] and Wolkotte [16]), possibly with time division multiplexing (TDM), (e.g. Æthereal [5, 6, 9, 14] and Nostrum [11]). The T-CREST NOC is based on the time division multiplexing principle, and this choice is motivated by the simple, straightforward and compositional approach to WCET analysis and by a simple and area-efficient router design.

As the NOC is the communication backbone that connects processor cores and memories, its design is influenced by the programmers model of the memory that is distributed among the cores, and thereby related to the design of the processor cores and the memory controller under development for the T-CREST platform. Discussions of these issues have spawned ideas for a novel and promising micro-architecture of the network interface; a design that integrates the DMA controllers, which are normally part of the processor cores, into the TDM slotting mechanism of the network interface. This new Network Interface(NI) micro-architecture avoids buffering and flow control; resources that is reported to account up to 85 % of the hardware area in existing designs [13].

Because of its simplicity we have developed a synthesizable register transfer level description of the new NI. Using this RTL model of the NI, and an RTL model of a 3-stage pipelined router, we can build a structural VHDL simulation model of the entire NOC. The model is parametrized in the number of processors and the number of virtual circuit connections (TDM slot tables, number of DMAs, etc.), and the model is clock-cycle and bit-true at the interfaces towards the processor cores.

This deliverable is structured as follows. In Section 2 we first introduce the T-CREST platform in order to provide the context for the design modeling of the NOC. Section 3 provides some background and related work on NI design. Following this, Section 4 presents the new area-efficient NI design with some preliminary synthesis results and a description of the router design used in the NOC. Section 5 presents the VHDL simulation model in more detail, and Section 6 provides some practical information on where and how to obtain the code, how to build a simulation model and how to run it. Section 7 revises the requirements of T-CREST on this work package, commenting whether and in which extend they are fulfilled. Finally section 8 concludes the report.

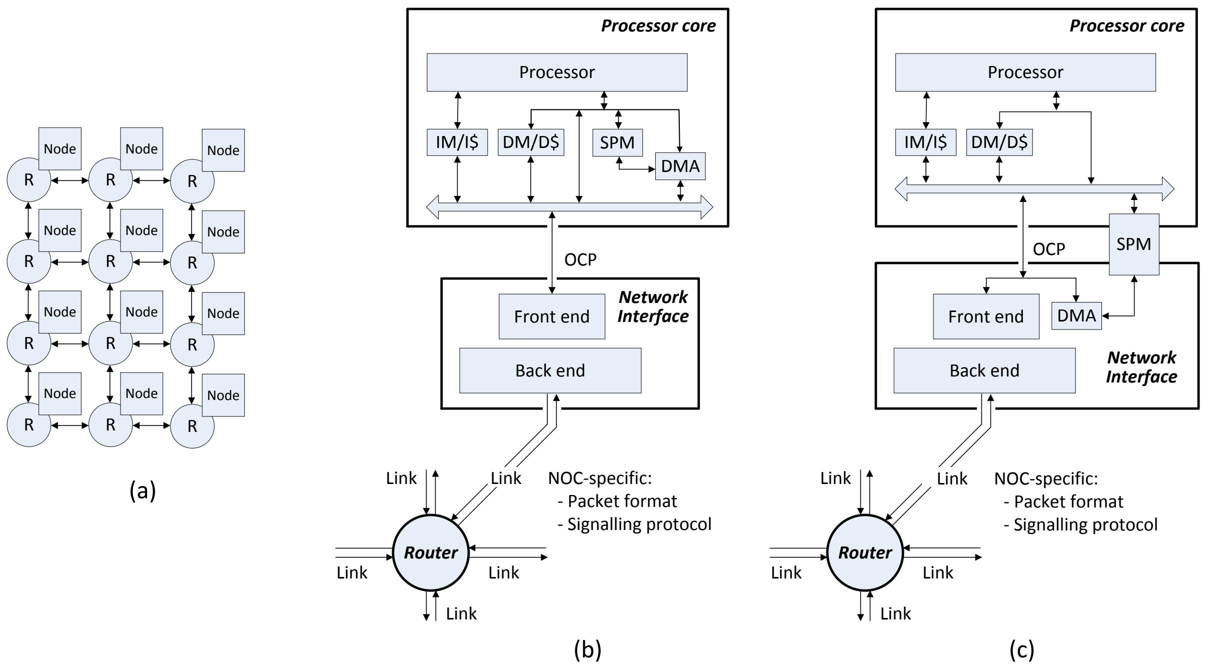

Figure 1: (a) An example NOC-based MPSoC using a 2D mesh NOC topology. (b) Details of a processor core and a network interface. The processor core consists of a CPU and local caches and scratch pads memories (SPM) and one or more DMA controllers. (c) Details of a processor core and a network interface in the T-CREST platform. The DMA controllers are moved into the NI and the two-ported SPM is used to bridge the clock domains.

2 The T-CREST Platform

The hardware platform currently under development in the T-CREST-project is illustrated in Figure 1. Figure 1(a) shows a generic NOC-based MPSoC using a 2D mesh NOC topology, and figure 1(b) shows a simplified view of a processor core and the network interface (NI) that connects it to the packet switched network, as they are expected to be in a general design. The packet switched NOC consists of a structure of routers and links – often connected to form a 2D mesh topology or a torus topology. A NI may be connected to any router port and typically a router has one or more NIs connected to it. A processor node contain local instruction and data caches and local scratchpad memories (SPM). One of the nodes in the platform will be a memory controller providing access to an off-chip SDRAM memory. The scratchpad memories in the processor nodes and the off-chip SDRAM memory are all mapped into a single address space. Each processor node will have a number of DMA controllers that are capable of performing block read and write transaction targeting the external SDRAM memory as well as scratchpad memories in remote nodes. DMA-driven block-writes may be used to support message passing. The design of the T-CREST NI and and the connection to the processor core is shown in Figure 1(c) and will be explained in detail in the following sections.

As prescribed in the document "D 1.1 Evaluation Requirements" the T-CREST NOC must support two categories of communication: (i) Processor to processor communication implemented as DMA-driven transfer of data-blocks among the scratch pad memories (SPMs) in the processor nodes. (ii) Processor to shared memory communication implemented using processor driven read/write transactions or DMA-driven block transfers towards the memory controller. The NOC discussed in this deliverable is intended for and optimized for the processor-to-processor communication. It may be used for processor to off-chip memory communication as well, but this may not be efficient and we are currently investigating the use of two separate NOCs in the T-CREST platform: An all-to-all NOC for the processor-to-processor communication using the new TDM-based NI and an all-towards-one network supporting communication between the processor core and the memory controller that administers access to the off-chip SDRAM.

To an application programmer the processor-to-processor NOC offers (configurable) point-to-point channels between any two cores. In a typical application the "connectivity" between processor cores is sparse and the schedule period correspondingly short, but as demonstrated in [12] even a fully connected topology can be supported by a TDM schedule with a modest period.

In order to simplify integration of different IP-cores, and in order to cope with the back-end timing-closure problems of modern silicon technologies, the T-CREST NOC will support a globally-asynchronous locally-synchronous (GALS) timing organization. More specifically the envisioned T-CREST platform will allow independently clocked cores (processor nodes, memory controller and IO-units) and its NOC will use mesochronously clocked network interfaces and a fully asynchronous packet-switched structure of routers and links. These issues are important for the final implementation, but are transparent to the cores that communicate across the NOC, and the simulation model excludes these issues – the simulation model corresponds to a simple synchronous circuit.

3 Background and Related Work

This section provides some background on the design of network interfaces in general and it presents and discusses some related work on TDM-based on chip networks, e.g., the aelite NOC [7] and details of the NI for this network.

3.1 TDM-based Networks-on-Chip and Aelite

The T-CREST NOC is based on time-division multiplexing (TDM) and is rooted work performed by T-CREST partner TU Eindhoven [5, 7]. TDM supports time-predictability in a straightforward way, and the routers are extremely simple and efficient. The 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 arbitration, flow control, and buffering. In this way the routers and links form a simple, switched, pipelined circuit. In each time-slot/clock cycle a router simply forwards data (i.e. flits) from its input ports to its output ports according to the routing information in the packet headers. From a hardware point of view it is a very simple design, and it is interesting to note that an aelite router is a factor of 10 smaller than for example a MANGO router [2] due to its smaller crossbar and its lack of arbitration, flow-control and especially virtual channel buffers.

In the T-CREST project we will develop a fully-asynchronous implementation of an aelite-style router but this is beyond the scope of this deliverable.

3.2 Design of Network Interfaces in General

NIs for different NOCs share some core functionality, but there are also substantial differences which follow from the fact that the NIs encapsulate the specifics of the underlying packet switched NOC. A general treatment of the topic as well as descriptions of some specific NI designs may be found in [1].

A Network Interface is generally divided into a *front-end* and a *back-end* as shown in Figure 1(b). The alternative terms *shell* and *network interface* are used in the Æthereal and aelite NOCs [5]. Towards the attached cores the front-end provides one or more ports implementing standard bus-style read-write transaction interfaces. The NI front-end transforms these transactions into some form of connection oriented streaming of packets. As elaborated in [1] this corresponds to the session layer in the seven-layer OSI reference model. The NI back-end implements the lower OSI-layers (transport layer, network layer and data-link layer). This involves data packetization and routing related functions that are specific for the specific NOC that is used.

Reliable data-link level communication typically calls for buffering and flow-control – often implemented using some form of credit-based scheme. Read or write transaction involving larger amounts of data may need to by transmitting as a sequence of smaller request and/or response packets whose size fits the TDM slots in the NOC. This splitting and re-assembly is also handled by the back end.

3.3 The Network Interface for the TDM-based aelite Network-on-Chip

In TU Eindhoven's aelite NOC, a NI connects one or more cores to a single router port. The NI front-end comprises one or more shells – one for each port towards the attached core. The NI back-end consists of a single network interface, which contains separate FIFO buffers for each shell. An end-to-end circuit has a port and a shell towards each of the cores it connects and corresponding FIFOs in the NI's back end use a credit-based mechanism to avoid overflowing the FIFO in the receiving end.

A specific aelite NOC instance has been designed for a TV-set platform [10], and a discussion of its implementation cost is provided in [8, 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. The extra shell is the port used to configure the NOC.

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 accounts for 85 % of this area. This figure assumes a flip-flop based buffer design. By using custom-layout FIFO-buffers the area can be halved. In our understanding it is the large number of FIFOs (one in each end of an end-to-end circuit connection) rather than the the total size of 24 KB that results in the large area. The total area of the 6 routers is only 0.08 mm^2. This corresponds to only 1.5 % of the NOC, and it shows that the routers are indeed very efficient and that all the complexity of the NOC is in the NIs.

In addition to the TV-set design reported above, aelite is also used in the CompSoC platform [7]. Processor nodes in this platform contain DMA controllers and local memories (but no caches). Some blocks of of memory are designated for communication purposes. These communication memories are dual-ported, and the DMA controllers are used to implement message passing by pushing data into the NI, across the network, and into a communication memory in the target processor core.

Clock domain crossing in aelite is done between the front-end (the shells) and the back-end (the network interface) using bi-synchronous FIFOs, and the NOC is typically synchronous or mesochronous.

4 NOC Architecture for the T-CREST Platform

Figure 1(c) shows the main features of a processor node and the NI that connects it to the packet switched NOC. The processor node includes some local memory (caches and explicitly managed scratch pad memories) and a DMA controller. In the T-CREST platform – as well as in the CoMP-SoC platform developed by TU Eindhoven [7] – the DMA controllers are intended to implement background DMA-driven block transfers from the local SPM into the SPM of a remote processor node. This effectively implements message passing. Our new NI micro-architecture, see Figure 4, pulls the DMA controllers into the NI, and in this way we avoid the need for flow control and buffering. The routers in the NOC follow the aelite design and are merely switching input to output ports and forward data through a pipeline. In the following we motivate and explain the new NI architecture as well as the router architecture, the two elements consisting the NOC.

4.1 Baseline Observations

The very essence of time-division-multiplexing is that it avoids buffering, flow-control, and dynamic arbitration in the implementation. Ideally this implies that it should be possible to transfer data all the way from the local memory (SPM) in one processor node and into the local memory (SPM) of another processor node without any buffering, flow control, and dynamic arbitration. The routers in the aelite NOC enjoy this simplicity, but the NIs in aelite do not. The essence of the problem is to the following: Data that is communicated over an an end-to-end virtual circuit is transferred through the NI in the source end, across the packet switched NOC, and across the NI in the destination NI. This typically involves a number of hops, not just from router to router, but also inside the NIs due to the layered implementation 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. These timing uncertainties may result from clock domain crossings, or – as discussed in [8, Sec. 2.4.2] – they may result from issues related to arbitration granularities. Our new NI design completely avoids these problems.

The overall functionality of the NI is to provide message passing send-transactions across end-to-end virtual circuits, and the NI implements this using DMA controllers that perform a sequence of write-transactions during the TDM slots assigned to the end-to-end circuit. Each write transaction is mapped into a packet which is sent into the NOC according to the global TDM schedule. Our key idea is to move the DMA controllers from the processor nodes into the NIs and to use the already dual-ported communication memories for clock domain crossing. This avoids the need for buffering, it allows the DMA controllers to directly deliver the payload data to outgoing packets, and it opens

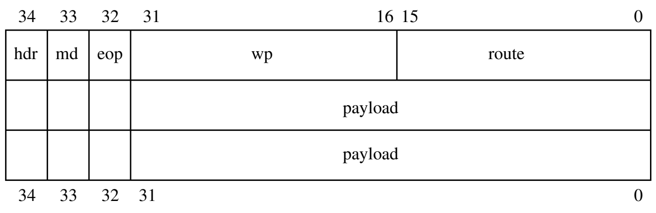

Figure 2: Packet format of the NoC.

for a very interesting and efficient table-based implementation of the DMA controllers. This novel architecture, as shown in Figure 4 is a key contribution of this deliverable and will be described in detail in the following Sections.

An application programmer may want to concurrently send messages to several other processors. In the aelite-based CompSoc platform there is one DMA controller per connection/channel [7]. An alternative, which is only possible in a best-effort NOC, and in applications where latency is not critical, is to send the messages in sequence using a single DMA controller. In a best-effort NOC this will make full use of the NOC bandwidth. In a TDM based NOC however, all communication channels are assigned some slots in the TDM schedule and in order not to waste bandwidth, 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 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.

Finally we mention that our aim is a GALS-style design allowing independently clocked processor cores. The TDM scheme requires that the NIs emit and absorb flits/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 packet/or 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. This is future work and will be completed in following tasks.

4.2 Micro-architecture of the TDM router

Routers in the T-CREST project follow the aelite router design [8]. They are simple 3-stage pipelined switches, matching the 3-flit packet size. A packet consists of 3 flits, a header flit and two data-flits each holding one 32-bit word. The header consist of the route (16 bits) and the local address in the target SPM (16 bits) and they are directly provided by the DMA table. The choice of sending two 32-bit words in one packet is arbitrary and represents a compromise between bandwidth utilization and a desire to have short TDM-slots and thereby a short schedule period. Each flit has additionally 3 bits for control information, to encode the header or payload flits and the valid packet. The 35-bit wide packet format is shown in Figure 2.

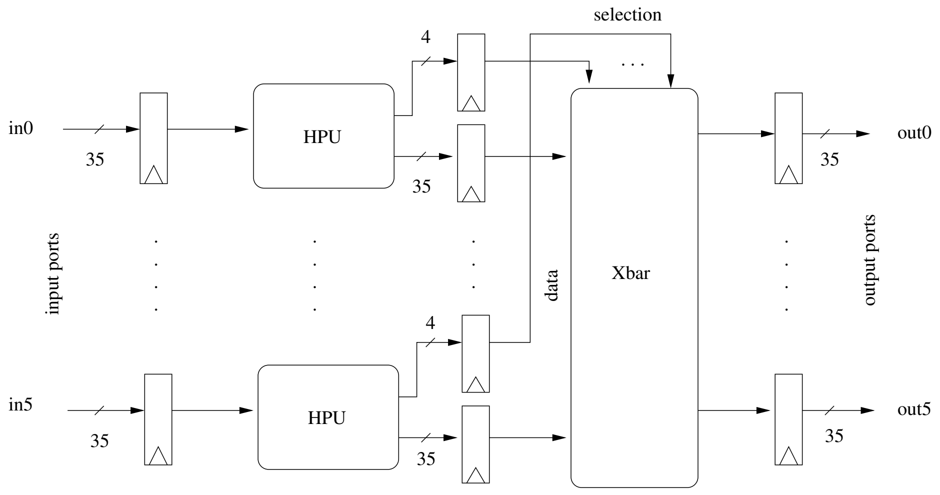

Figure 3: Block diagram showing the micro-architecture of the TDM router.

They are 5-ported, forwarding the data from 5 incoming ports (north, south, east, west, local) to 5 output ports. No buffering or any control is included in the routers and they do not store routing information, which makes them very simple and lightweight. A block diagram of the router is shown in Figure 3. The 3 pipeline stages are: (1) link traversal, (2) Header Parsing Unit (HPU) and (3) crossbar. The HPU serves to decode the route in the header flit, forward the selection to the next stage and shift the control bit in place. The selection is saved when a header flit is fed in the HPU and is kept for three cycles throughout the time slot. The crossbar is switching the data to the appropriate output port based on the routing information.

To deal with timing synchronization issues that may come up along a path, mesochronous or asynchronous clocking is needed between the routers and between routers and NIs. In the current simulation model synchronous timing is used but the aim of this project is to include completely asynchronous routers. An asynchronous implementation would keep the same basic pipeline, incorporating asynchronous handshaking and synchronization techniques to cover for uncertainties on the links.

4.3 Micro-architecture of the T-CREST Network Interface

The key elements of the micro-architecture, Figure 4, is 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 schedule-period. The slot counter indexes a *slot table* whose entries consist of a valid bit and an index into the DMA table. The valid bit indicate whether or not a packet is to be sent in the current time-slot. If the valid bit is true, the entry also holds a pointer to the relevant entry in the DMA table. It is thus possible to assign several slots in the TDM period to the same end-to-end circuit by indexing to the same DMA table entry several times. In this way different amounts of bandwidth can be assigned to different end-to-end circuits. 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 corresponding to a write-word transaction is sent across the NOC.

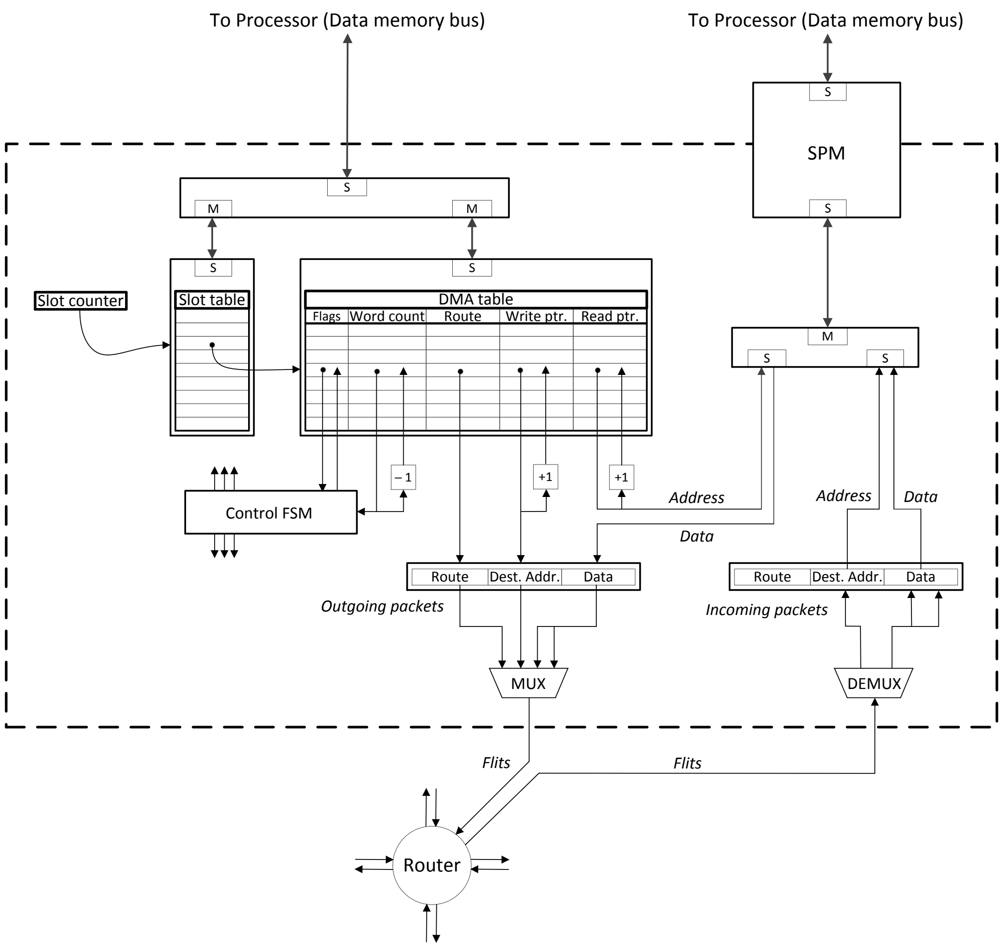

Figure 4: Block diagram showing the micro-architecture of the new NI for the inter processor communication NOC.

The interfaces denoted "M" and "S" are master and slave ports supporting OCP style point-to-point read/write transactions.

A TDM time slot corresponds to the time it takes to transmit a packet into the NOC (or to receive a packet from the NOC). In the implementation described in Section 4.4 a packet consists of 3 flits and consequently a TDM slot is 3 clock cycles. During a slot-period it is thus possible to access the SPM, the slot table and the DMA table 3 times, which is more than enough to support the different needs. Other implementations of the architecture may use different packet formats and/or perhaps double clock the NOC.

A consequence of the new architecture is that transactions where the processor accesses the entries in the DMA table crosses the clock domain boundary, see Figure 4. This means that these transactions (or at least the read transactions) will suffer from the latency of the synchronization [4]. Assuming dual flip-flop synchronizers a read-transaction may experience a 2-4 cycle added latency. As setting

NI design size		Resources		
Slot Period	**DMAs**	**LUTs**	**FFs**	**BRAM [bit]**
32	32	380	166	2240
64	64	385	167	4544

Table 1: Area figures for a selection of NI implementations when synthesized for an ALTERA Cyclone II EP3C70 FPGA.

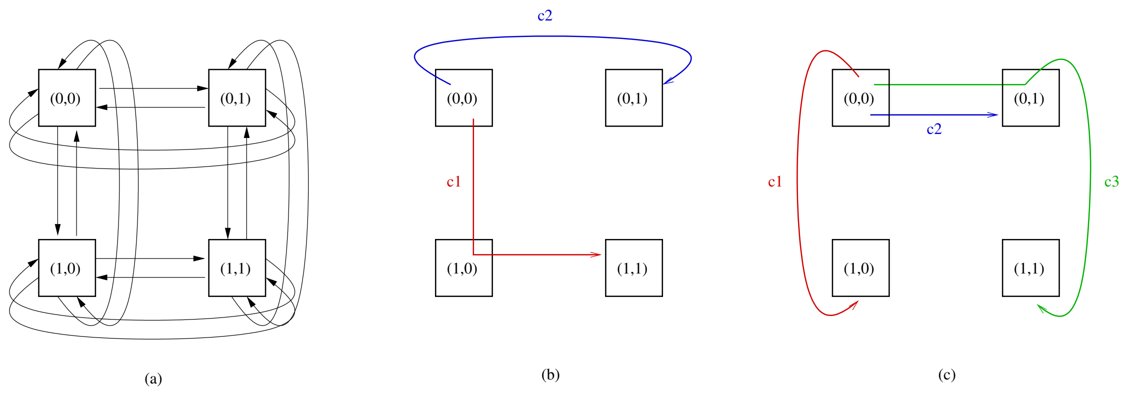

Figure 5: Test cases for the simulation model. (a) 2x2 bi-torus NOC simulated, (b) Test-case0, (b) Test-case1.

up DMA transfers is likely the be rare events compared to DMA initiated read or write transactions this is a small overhead.

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

4.4 Implementation and Results

To evaluate the new NOC design we have developed a parametrized VHDL description of a NOC using the new NI architecture and the 3-stage pipelined aelite-style router.

To evaluate the new NI design and get some resource estimation we have synthesized the parametrized VHDL description of the NI-design. The NI description has been synthesized for an ALTERA DE2-70 FPGA board with a Cyclone II EP3C70 chip. 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 has been synthesized [13]. A few results are shown in Table 1. The table lists the amount of hardware resources used: the number of (4-input) look-up-tables, the number of flip-flops, and the size of the block RAM used to implement the slot table and the DMA table. The figures show that the size of the logic is constant across the different NI-instances: 380-385 LUTs and 166-167 FFs. The amount of BRAM matches the size of the slot table and the DMA table.

5 The VHDL Simulation Model

To evaluate the new NI design we have developed a parametrized RTL-level VHDL description of a NOC using the new NI architecture and the 3-stage pipelined aelite-style router. The model includes the VHDL description of the NI and the router as they are described in the previous section, connected in a NxN bi-torus topology. Additionally, it includes VHDL description for the SPMs. The dimensions of the NoC are defined as parameters in a VHDL package description, adjusting sizes such as the dimension N of the network,the length of the slot period, the number of DMA controllers etc.

Towards the processor data bus the NI offers a small subset of OCP transactions (read single word, write single word). The processor has access to the slot table (for initialization purposes) and to the DMA table. Each DMA table entry is mapped into a sequence of words in the processor address map. 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.

In order to allow reading of two words for an outgoing packet and writing of two words from an incoming packet in a slot period of 3 clock cycles, the SPM interface towards the NI support double-word read and write transactions. For this reason the pointers in the DMA table are incremented or decremented by two. Many variations are possible including the use of separate send and receive SPMs and different packet formats.

In more detail, each network node is described as an entity consisting of one NI instance, a router instance and an SPM instance. The NI entity contains an instance for each main structural component as described in the previous section i.e. a slot counter entity instance, a slot table entity instance, a DMA table entity instance and some additional logic. The slot table and the DMA table entities are described as RAM memories, with some additional controlling logic in the DMA table entity. The router entity contains entity instances for the HPU and the crossbar. SPM entity is described as a dual-ported RAM memory. The network node provides two interfaces to the processor and two ports to the network. One interface to the processor is an OCP interface and it is used for read and write transactions, initiated from the processor in order to configure and get control information from the NI tables. The second interface to the processor is for SPM read and writes. The two network ports (input/output) are flit-wide ports for the serial transmission of flits to/from the network.

The slot schedule as well as the DMA controllers are configurable by the processor, as mentioned before. It is done through the OCP interface as a series of OCP writes to a specified address in the address space. In the current simulation model the address space mapping is defined in a VHDL package description. The slot table corresponds to a continuous address space stored as a block-aligned array of words starting at address x10000000. The route field of the DMA controllers is also placed in a separate continuous address as a block-aligned array of words starting at addresses x18000000". The rest of the configuration of the DMA controllers starts at address x2000000 and is placed as a two-word block-aligned array. We envision that the slot table and the route field of the DMAs are going to be accessed only in protected mode, for configuration purposes, while the rest of the DMAs in normal mode.

To test the design, a square 2x2 bi-torus network, as shown in Figure 5(a), was simulated with different configurations concerning the slot schedule and the DMA controllers. An environment was developed feeding test vectors to the input ports of the NI and the SPM, simulating the behavior of a processor. The simulation process contains a full usage scenario of the NOC, as described next.

Initially, some data are written in the SPM of a node through the processor-to-SPM interface. A schedule, assigning slots to DMA controllers, is configured into the slot table of the node through the processor-to-NI interface, as a series of write transactions. The DMA controllers are also initialized as a series of write transactions through the same interface. First, the route is programmed and then the control information of the DMA controller. As soon as the "start" bit of a DMA controller is set, the operation of the controller starts on the next slot assigned to this controller. On the slot assigned to this DMA a packet will be emitted from the NI to the switching network, carrying the routing information and the data from SPM. The packet follows the defined route and is eventually written in the destination SPM. This is checked by a read transaction from the processor-to-SPM interface on the destination node.

The environment simulated configures a slot schedule of 8 time-slots and 4 DMA controllers in each NI. Three test-cases are provided with the simulation model. Test-case0 is shown in Figure 5(b). NI of node (0,0) is configured with a schedule including invalid slots and slots pointing to two out of the four NI DMA controllers (DMA0 and DMA1). The two DMA controllers are programmed to handle two communication channels with different routes and destinations:

Channel *c1*: (0,0)S -> (0,1)E -> (1,1).

Channel *c2*: (0,0)W -> (0,1).

At the destination nodes, (1,1) and (0,1), the data is read by the processor-to-SPM interface after sufficient time has passed. Test-case1 is shown in Figure 5(c). NI of node (0,0) is programmed to host 3 channels.

Channel *c1*: (0,0)N -> (1,1)

Channel *c2*: (0,0)E -> (0,1)

Channel *c3*: (0,0)E -> (0,1)N -> (1,1)

Test-case2 simulates all-to-all communication. All NIs are programmed with the same DMA configuration as in Test-case1, holding 3 channels each, with the same routing details. Each node holds different SPM data, that is transferred to all other nodes. At the end all nodes contain the same SPM data received from the other nodes.

6 Accessing the Source Code

The source code along with a simulation testbench is provided through the t-crest repository via the `git` source code management tool:

```
https://github.com/t-crest/t-crest-noc
```

6.1 Requirements

To compile and simulate of the t-crest NOC the following tools are needed:

- A Unix like environment with `git`, `make`, and a C/C++ compiler, such as: Linux, Mac OSX, or cygwin/Windows
- A recent version of `cmake`
- ModelSim for simulation (the free version from Altera is good enough) [1]

6.2 Retrieving and Running the Source Code

The VHDL description model of the T-CREST NOC can be retrieved as follows:

```
git clone git://github.com/t-crest/noc.git
```

or downloaded as .zip file from GitHub:

`https://github.com/t-crest/t-crest-noc/zipball/master`

The simulation of the t-crest NOC is `make` based.

A plain, and simple

```
make
```

will: (1) compile the VHDL design, (2) simulate the model along with the environment, (3) generate a waveform with the signals showing the operation.

Following `make` targets are available to simulate the different test-cases that are described in Section 5 or clean up from temporary files:

`make test0`	simulate test-case0
`make test1`	simulate test-case1
`make test2`	simulate test-case2
`make clean`	remove (most) temporary files

The `Makefile` is intended to support: Linux, a cygwin environment under Windows, and Mac OSX. Under Mac OSX the Windows version of ModelSim is supported via `wine`.

[1] `https://www.altera.com/download/software/modelsim-starter`

7 Requirements

In this section all requirements in aspect CORE and scope NEAR from Deliverable D 1.1, which are relevant for the network-on-chip work package, are listed. NON-CORE and FAR requirements are not repeated here. The requirements are followed by a comment to what extent they are fulfilled by the simulation model.

N-3-006 The NoC shall be time-predictable (i.e. temporal bounds shall be provided for the time required by processing nodes to exchange data with off-chip main memory, or with remote nodes' SPMs).

The current simulation model is time-predictable.

N-3-043 The NoC shall support GALS style design.

Not applicable in the simulation model. Will be supported in future tasks.

N-3-044 The NoC shall provide communication channels between processing nodes and between processing nodes and main memory.

The current simulation model provides communication channels between processing nodes. Communication to main memory is not supported yet, according to the planning reported in the M9 Interim Project Report.

N-3-045 The NOC should allow for data transfers in blocks handled by DMA controllers.

Data transfers in blocks handled by DMA controllers are fully supported.

N-3-046 The NOC shall provide the processing nodes with mechanisms for pushing data to SPM in remote nodes and to the memory controller.

The simulation model supports pushing data from processing nodes to SPM in remote nodes.

N-3-047 The NOC shall provide the processing nodes with mechanisms for pulling data from main memory.

Not yet supported, according to the planning reported in the M9 Interim Project Report.

N-3-048 The NOC should provide the processing nodes with mechanisms for pulling data from SPM in remote nodes.

Not yet supported. Requires a simple extension of the NI.

N-3-049 The NOC shall be configurable at initialization.

The current simulation model is configurable at initialization.

N-3-051 The NOC should provide flow control mechanisms to end-users in order to prevent it from blocking.

Flow control is not needed as the new NI uses a global TDM-schedule which prevents the NOC from blocking.

N-3-052 The NoC shall support at least 64 tiles.

The current simulation model supports up to 64 tiles.

N-3-053 The NoC shall support DMA driven block write from local SPM to remote SPM.

Block write from local SPM to remote SPM is fully supported through DMA controllers.

N-3-054 The NoC should support DMA driven block read from remote SPM into local SPM.

Not yet supported. Requires a simple extension of the NI.

N-3-055 The NoC shall support DMA driven block write from local SPM to off-chip memory (memory controller).

Not yet supported, according to the planning reported in the M9 Interim Project Report.

N-3-056 The NoC shall support DMA driven block read from main memory into local SPM.

Not yet supported, according to the planning reported in the M9 Interim Project Report.

N-3-057 The NoC shall support processor driven write (from SPM and caches) to off-chip memory (memory controller).

Not yet supported, according to the planning reported in the M9 Interim Project Report.

N-3-058 The NoC shall support processor driven read from main memory into local memories (SPM and caches).

Not yet supported, according to the planning reported in the M9 Interim Project Report.

N-0-063 The NoC controller shall contain a performance counter which can be read out for performance analysis.

Not applicable in the simulation model. Will be supported in future tasks.

N-2-011 The processor may have several read or write requests outstanding. The NoC shall not reorder those read or write requests from the processor.

The simulation model follows the TDM scheme, thus not allowing for any reordering.

N-6-040 Any access to a processor-external resource (i.e.: memory, NoC) shall execute in bounded time (depending on resource and access type).

NOC simulation model contributes bounded latency.

N-4-020 The NoC shall be time-predictable; temporal bounds on the time to produce and consume outstanding requests, to execute configuration code, and to transport configuration settings to the DRAM controller shall be provided.

Serving of requests is time-predictable. Reconfiguration is a processor local action, therefore, time-predictable.

8 Conclusion

In this deliverable we presented the simulation model of the time-predictable NOC and a novel design for a NI developed for this project. This deliverable focuses on the design of processor-to-processor communication interconnect, as it is the most demanding in terms of communication and analyzability. To provide analyzability and bounded latency and bandwidth guarantees TDM scheme was adopted. Based on this scheme a novel, area-efficient NI architecture was explored. This new NI design was developed and implemented proving to be very efficient in terms of resources. A simulation model of the interconnect was developed using this NI and a TDM router and verified through a number of test cases. The specific model is time-predictable and analyzable and can be used to connect multiple processor instances for further development within the T-CREST project.

References

[1] D. Berozzi. Network interface architecture and design issues. In G. DeMicheli and L. Benini, editors, *Networks on Chips*, chapter 6, pages 203–284. Morgan Kaufmann Publishers, 2006.

[2] 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)*, pages 1226–1231. IEEE Computer Society Press, 2005.

[3] T. Bjerregaard and J. Sparsø. A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-chip. In *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 34–43. IEEE Computer Society Press, 2005. (Best paper award).

[4] R. Ginosar. Metastability and synchronizers: A tutorial. *IEEE Design & Test of Computers*, 28(5):23–35, 2011.

[5] 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*, pages 306 –311, June 2010.

[6] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5):414–421, Sept-Oct 2005.

[7] 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*, 14(1), 2009.

[8] Andreas Hansson and Kees Goossens. *On-chip interconnect with aelite / Composable and predictable systems*. Springer, 2011. Embedded systems.

[9] Andreas Hansson, Mahesh Subburaman, and Kees 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)*, pages 250–255, 2009.

[10] Peter Kollig, Colin Osborne, and Tomas Henriksson. Heterogeneous multi-core platform for consumer multimedia applications. *Proceedings -Design, Automation and Test in Europe, DATE*, pages 1254–1259, 2009.

[11] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. The Nostrum backbone - a communication protocol stack for networks on chip. In *Proceedings of the VLSI Design Conference*, pages 693–696, Mumbai, India, January 2004.

[12] 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)*, pages 152–160. IEEE Computer Society Press, May 2012.

[13] J. Sparsø, E. Kasapaki, and M. Schoeberl. An area-efficient network interface for a tdm-based network-on-chip. *Submitted and under review*.

[14] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens. A TDM NoC supporting QoS, multicast, and fast connection set-up. *Proceedings - Design, Automation, and Test in Europe Conference and Exhibition*, pages 1283–1288, 2012.

[15] Daniel Wiklund and Dake Liu. SoCBUS: Switched network on chip for hard real time embedded systems. page 78a. IEEE Computer Society, 2003.

[16] Pascal T. Wolkotte, Gerard J.M. Smit, Gerard K. Rauwerda, and L. T. Smit. An energy-efficient reconfigurable circuit switched network-on-chip. page 8 pp., April 2005.