T-CREST

TIME-PREDICTABLE MULTI-CORE ARCHITECTURE
FOR EMBEDDED SYSTEMS

Project Number 288008

D 3.4 Report documenting hardware implementation of the self-timed NOC

Version 1.0
23 September 2013
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	24 August 2013
0.3	Internal Revision	31 August 2013
0.9	Draft sent to T-CREST partners	9 September 2013
1.0	Final	23 September 2013

Executive Summary

This document describes the deliverable *D 3.4 Report documenting hardware implementation of the self-timed NOC* of work package 3 of the T-CREST project, due 24 months after project start as stated in the Description of Work.

The document presents the hardware implementation of the self-timed network-on-chip (NOC). The self-timed NOC refers to the network that connects the processors and handles the inter-processor communication. The implementation involves a new area-efficient network interface(NI) design and a new asynchronous router design. The NOC is described as parameterized register transfer level VHDL, from which any network size can be instantiated. For verifying the implementation we have instantiated a 2x2 bi-torus network. The gate netlist is produced through synthesis in a $65\,nm$ CMOS STMicroelectronics Technology, using Synopsys Design Compiler.

The source code and the gate-netlist along with the corresponding scripts for synthesis and simulation are provided through the T-CREST repository via the `git` source code management tool. This document includes information on how to build and run the VHDL simulation model and the corresponding gate-netlist. In addition, it provides synthesis scripts that define the implementation parameters and constraints. The functionality has been verified by running a number of tests that are available as part of the source code.

1 Introduction

Deliverable D3.4 is a report providing a description of the hardware implementation of the self-timed Network-on-Chip (NOC) as was presented in deliverable D3.3. The self-timed NOC refers to the network that connects the processors and handles the communication between processors, i.e. *"communication network"*. A network providing access to main memory will be implemented in WP4 and is out of the scope of this deliverable.

The implementation of the NOC follows the simulation model that was presented in deliverable D3.2, using asynchronous routers to agree with the self-timing requirement. A new router design is introduced that is fully asynchronous. The Network Interface (NI) is modified to fit the connection with the new asynchronous router. The NOC is described as a parameterized structural RTL-level VHDL design, using the new NI and router designs. From that description a synthesized gate-netlist of any size can be derived. For deliverable D3.4 a 2x2 instance of the NOC has been synthesized in a 65nm STMicroelectronics technology library.

This deliverable is structured as follows: In Section 2 we give an overview of the T-CREST platform and its timing organization to set the context for the implementation of the self-timed NOC. Section 3 presents the architecture of the NOC and the micro-architecture of the router and the NI. This includes the new asynchronous router design and the NI architecture previously described in deliverable D3.2 [3] with some minor modification to fit the asynchronous router. Section 4 provides implementation details for the NOC and some evaluation results. Section 5 gives information on how to program and test the NOC. Information on how to obtain the code of the self-timed NOC, how to synthesize the gate-netlist and simulate the provided test cases is included in Section 6. Section 7 reviews the requirements of T-CREST related to this work package, commenting whether and to which extend they are fulfilled. Finally section 8 concludes the report.

2 The T-CREST Platform

The hardware platform of the T-CREST-project is illustrated in Figure 1. Figure 1(a) shows a generic NOC-based MPSoC using a 2D mesh NOC topology. It consists of a set of processor cores communicating through a packet switched structure of routers and links. The processor cores in Figure 1 are the Patmos processor [6] implemented for T-CREST platform. Each Patmos processor has some amount of local memory and it is connected to the network through a network interface (NI). The NI translates (core-specific) write transactions into (NOC specific) packets. The network of routers is a packet switched network that can be structured in any topology. For this deliverable we have chosen to build a bi-torus topology.

Figure 1(b) shows a simplified view of the processor core and the NI that connects it to the packet switched network. The packet switched NOC consists of a structure of routers and links. A NI is connected to a router port and each router has one NI connected to it. A processor core contains local instruction and data caches and local scratchpad memories (SPM). This work package (WP3) focuses on the "communication NOC" that accommodates the communication between processor cores. Access to an off-chip SDRAM memory is provided through a separate network connected to a memory controller. The scratchpad memories in the processor cores and the off-chip SDRAM

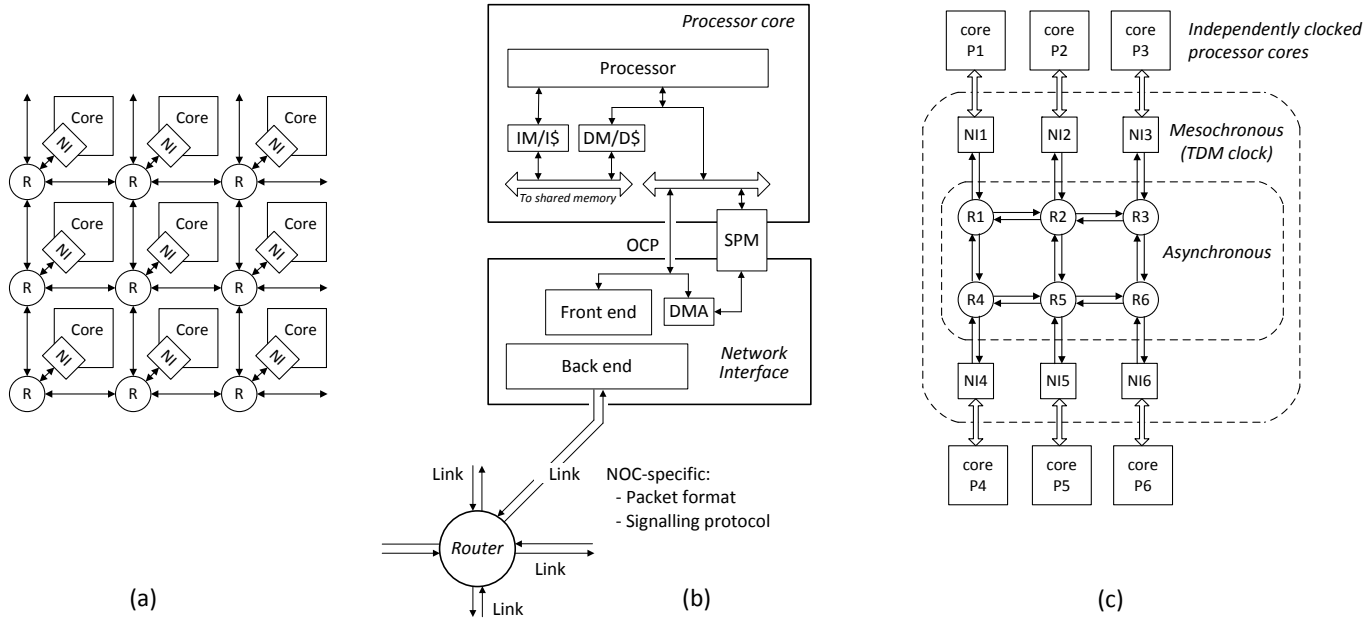

Figure 1: (a) An example NOC-based MPSoC using a 2D mesh NOC topology (b) Detailed view of a processor core and a network interface in the T-CREST platform. The DMA controllers are moved into the NI and the dual-ported SPM is used to bridge the clock domains (c) Timing organization of T-CREST platform.

memory are all mapped into a single address space. Traditionally, a processor core has a number of DMA controllers that handle transports of blocks of data. In the T-CREST platform the DMAs are pulled into the NI and are capable of performing block write transactions targeting the scratchpad memories in remote cores through the "communication NOC". DMA-driven block-writes are used to support message passing. The design of the T-CREST NI and and the connection to the processor core shown in Figure 1(b) will be explained in detail in the following sections.

In order to simplify the integration of processing cores with different frequencies, and in order to cope with the back-end timing-closure problems of modern silicon technologies, the T-CREST NOC supports a globally-asynchronous locally-synchronous (GALS) timing organization. More specifically the T-CREST platform allows independently clocked processor cores and its NOC uses mesochronously clocked network interfaces and a fully asynchronous packet-switched structure of routers and links. Mesochronous is the timing organization where each component is clocked with the same clock but some bounded skew can be tolerated, thus NIs are clocked by the same NI clock. The overall timing architecture of the platform is illustrated in Fig.1(c) – a GALS-style design using asynchronous routers, mesochronous NIs and (possibly) independently clocked processor cores.

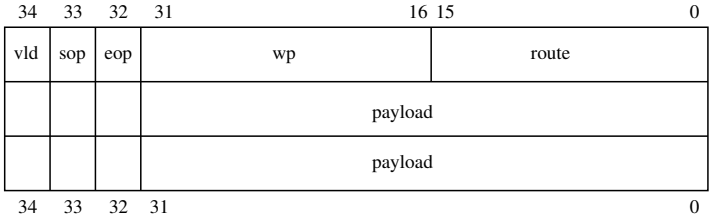

Figure 2: Packet format of the T-CREST NOC.

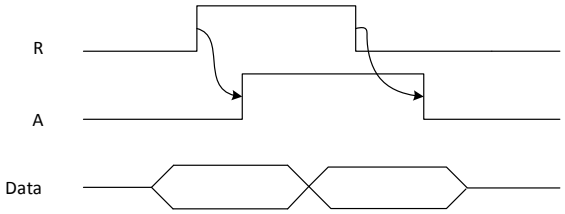

Figure 3: Handshake timing diagram.

3 NOC Architecture

This Section presents the basic concepts and the micro-architecture of the NI, and the asynchronous router. The NI is based on the VHDL simulation model that was presented in Deliverable 3.2 [3] and published in [9]. This design was extended to accommodate a phase delay in the incoming and outgoing packets from and to the network, as will be explained in Section 3.3. The router design is based on the synchronous 3-stage pipeline router that was presented in deliverable D3.2 [3]. The handshaking between the latches in the router uses the 2-phase bundled data protocol and uses the MOUSETRAP latch controller [7]. The design has been presented in [5], and is described in Section 3.2.

3.1 Basic Concepts of the Self-Timed T-CREST NOC

The routers for the T-CREST NOC are 5-ported, 3-stage pipelined switches. A packet consists of 3 phits, a header phit and two payload phits each holding one 32-bit word, as it appears in the packet format in Figure 2. The header phit consist of the route (route - 16 bits) and the write pointer (wp - 16 bits), i.e. the local address in the target SPM, and they are directly provided by the DMA table. Each phit has 3 additional bits for control information, to encode that the packet is valid (vld), the start (sop) and the end of packet (eop). The control information was adjusted from Deliverable D3.2 and the new packet format is 35-bit wide, as shown in Figure 2.

The synchronization in the routers is done based on a 2-phase bundled-data protocol. This is a non-return-to-zero (NRZ) protocol. A timing digram of the handshake can be seen in Figure 3. A handshake starts with a transition of request signal (R), either rise or fall, and end with the corresponding transition of acknowledge signal (A). Data need to be ready before the transition of request and need to be stable until the transition of acknowledge. In Figure 3 two handshakes are shown.

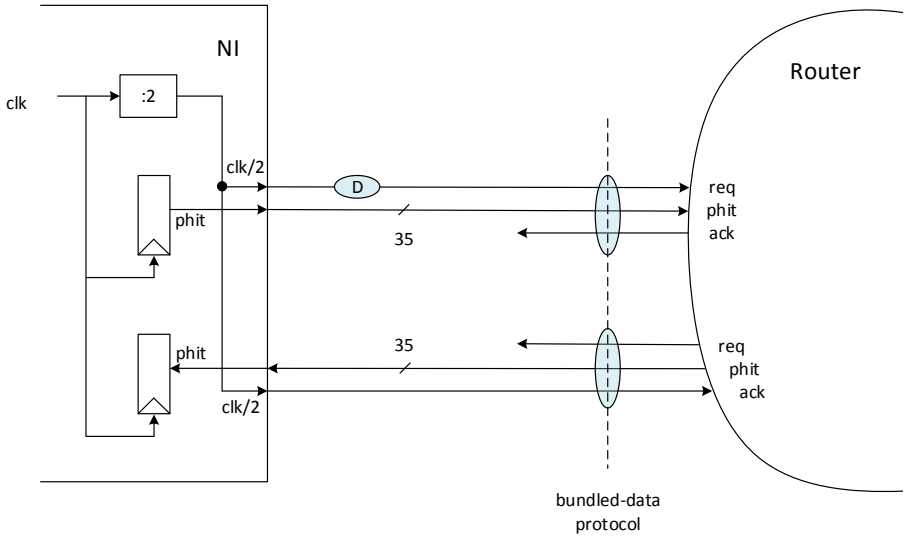

Figure 4: Connecting interface of synchronous NI with asynchronous router.

The NI is connected to the asynchronous router as shown in Figure 4. There is an incoming and an outgoing channel to/from the NI. The clock of the NI is divided by 2, in order to provide the request/acknowledge signal for the 2-phase handshake protocol. For the outgoing channel from the NI, the halved clock is connected to the request signal, while the acknowledge signal is ignored. A delay element is required in the request line, such that when the request arrives in the router, the phit coming from a clocked register is ready and stable. For the incoming channel to the NI the halved clock is connected to the acknowledge signal, while the request is ignored. This has the implied requirement that the network of routers is faster than the NI clock. As long as that this requirement holds, the NOC functions correctly. In the implementation results it is shown that this requirement is kept as handshake frequency of the routers is higher than the clock frequency of the NIs.

The basic idea of the NOC is that in one NI clock cycle, 1 phit is transmitted from the NI to the router and vice-versa. In the same way, in one handshake cycle, 1 phit is transmitted from a router's stage to the next. Since there is always a clock-tick or a handshake-tick, a phit must be transmitted in every tick. If there is no data to be transmitted, a void-phit is transmitted. That is a phit consisting of zeroes, identified by a 0 in the valid bit field. Thus, there is always a phit transmitted in a NI clock-tick or in a handshake-tick, being either a valid or a void phit.

The entire structure of routers and links must be reset to an initial state. Due to the synchronization point in the crossbar stage, as long as the NI are not inserting phits into the network, the initial state will persist when reset is de-asserted. Based on studies of dynamic wavelength [8], the handshake latches at the inputs and outputs of the crossbar are initialized to hold void phits. The handshake latches at the inputs are initialized to be transparent.

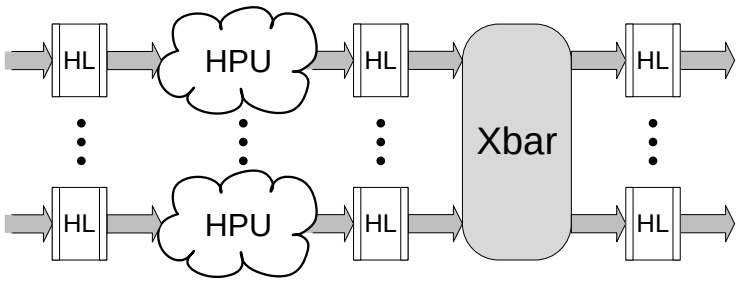

Figure 5: Block diagram of the TDM router.

3.2 Router Micro-Architecture

T-CREST routers are similar to clocked TDM routers but use handshake latches instead of registers. A block diagram of the router is shown in Figure 5. As described in [4] and as presented in Deliverable D3.2 report [3] the routers are 5-ported, forwarding the data from 5 incoming ports (north, south, east, west, local) to 5 output ports. No buffering or flow control is included in the routers and they do not store tables with routing information, which makes them very simple and lightweight. As shown in Figure 5, the 3 pipeline stages are: (1) link traversal, (2) Header Parsing Unit (HPU) and (3) crossbar. The HPU decodes the route from the header flit, forwards the selection to the next stage and shifts the control bits in place. The selection is latched 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.

A range of asynchronous router designs have been previously explored in [5]. As in all asynchronous circuits, handshaking is used among neighboring pipeline stages to control the transfer of data. In one full handshake cycle one token of data (i.e. phit) is forwarded from one stage to the next. In the T-CREST router design handshake latches are used in each pipeline stage instead of registers. A handshake latch is composed of a regular enable latch and a handshake latch controller [8]. The handshake protocol is a 2-phase (non return-to-zero) bundled-data handshake protocol. Delay elements are needed in the request signal paths to match the propagation delays in the combinational logic i.e. links, HPU and crossbar.

A detailed view of the T-CREST handshake latch appears in Figure 6. The latch controller used in the T-CREST NOC uses the MOUSETRAP controller described in [7]. This controller was chosen after implementation and comparison with the designs presented in [5], as it is very efficient and easy to implement. As shown in Figure 6, the MOUSETRAP controller implements a 2-phase bundled-data on the input/left side and one on the output/right side. Request (Ri/Ro) and acknowledge (Ai/Ao) signals are used in each side to implement the handshake with the neighboring controller. The controller consists of a latch and an XNOR gate. No special cells for asynchronous logic (C-elements) are used and no timing assumptions inside the controller need to hold for correct function. Furthermore, it is very efficient as is based on just a latch and a simple logic gate.

Additionally, we have extended the MOUSETRAP controller with a *"clock-gating"* scheme, as it was presented in [5], in order to reduce power consumption. The phits forwarded with the asynchronous handshaking may represent either valid-phits or void-phits. The valid bit (vld) in the packet format indicates whether a valid- or a void-phit is forwarded. In case of a void-phit the latch enable signal

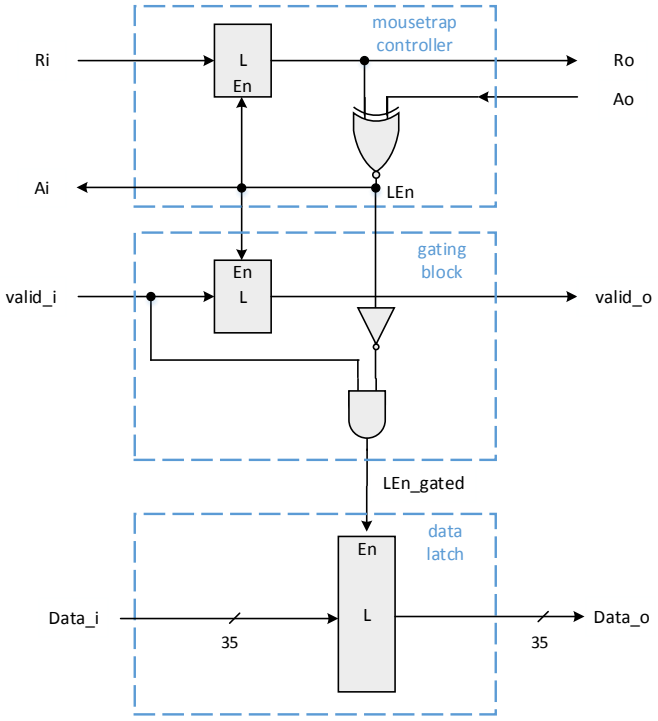

Figure 6: T-CREST handshake latch, consisting of three blocks: (i) the MOUSETRAP controller, (ii) the gating block and (iii) the normal enable data latch.

that controls all the data latches is disabled. To provide this functionality an additional gating block stands between the controller and the enable latch as shown in Figure 6.

3.3 NI Micro-Architecture

The micro-architecture of the NI is shown in Figure 7. As presented in Deliverable 3.2 [3] and published in [9], the key elements of the micro-architecture are the *slot counter*, the *slot table* and the *DMA table*. The slot counter is incremented in all NIs using the same (mesochronous) clock. The slot counter defines the slots in the TDM schedule-period. The slot counter indexes a *slot table*, where each entry consists 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 current time-slot. If the valid bit is true, the entry also holds a pointer to the relevant entry in the DMA table. An entry in the DMA table holds all the registers that are found in a normal DMA controller (control bits, a read pointer, a write pointer and a word count). Additionally, source routing requires the routing information, which is also included in the DMA entry.

The NI provides one interface to the processor, one interface to the SPM memory and two interfaces to the network. The interface to the processor is an OCP [1] interface and it is used to configure the DMA tables and get control information from the DMA tables. The second interface is for SPM

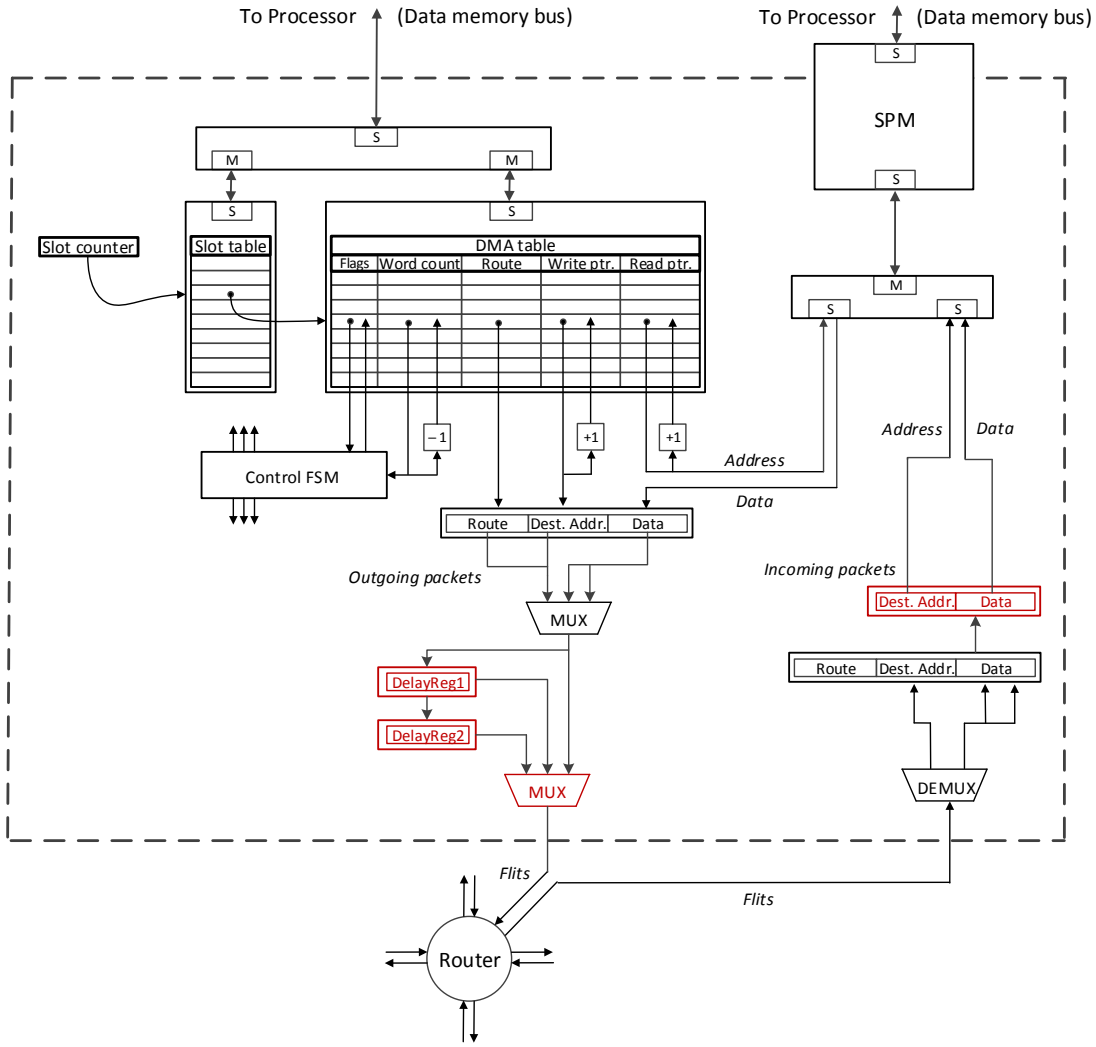

Figure 7: Block diagram showing the micro-architecture of the new NI for the inter processor communication NOC.

reads and writes. The two network ports (input/output) are phit-wide ports for the transmission of phits to/from the network.

This NI design requires some changes over the previous design described in [3] in order to accommodate the connection to the network of asynchronous routers. Asynchronous pipelines require a number of empty latches (i.e. bubbles) in order for data phits to be forwarded. Thus, routers cannot store one whole packet(3 phits) and a fine-grained scheduling is needed, at the level of phits. Since packets are organized in sizes of 3 phits, phit-scheduling results in packets scheduled with possible gaps of 0, 1 or 2 cycles in-between. The NI design was extended to accommodate a phase delay of either 0, 1 or 2 cycles in the incoming and outgoing packets from and to the network.

To integrate this functionality, changes were required in the incoming and outgoing path from and to the network. The new micro-architecture appears in Figure 7. The additional features appear in red. In the incoming part a buffer is added to hold the write address and data of arriving packets when the last phit of the packet arrives. The writing of data in the SPM is delayed such that the whole packet has arrived also considering the phase delay. In the outgoing path, an additional 2-bit field was added to the slot table to encode the phase delay (0, 1 or 2 clock cycles). Two single-phit buffers are added in order to support delaying the outgoing phit 1 or 2 cycles and based on the phase delay field of the slot table the corresponding phit is forwarded to the network.

4 Implementation Details

This section addresses the design flow and implementation details for the NOC along with the challenges faced when implementing asynchronous circuits using conventional EDA-tools.

The implementation of the NOC involves synthesis to a gate-level netlist. For the implementation the Register-Transfer-Level VHDL description of the NOC developed in Deliverable D3.2 was used. That is a parameterized NxN bi-torus network, supporting up to 64 nodes. The modified NI architecture and the asynchronous MOUSETRAP router were used as described in the previous section.

To get a gate-level implementation a 2x2 bi-torus network was synthesized, using a 65nm CMOS standard cell library from STMicroelectronics with HVT and SVT cells. Synthesis was performed using Design Compiler from Synopsys. The design was tested through simulation using ModelSim.

The gate-netlist implementation consists of 4 NI nodes and 4 routers. It was synthesized with an aim for speed constraining the clock of the design. The parts of the NI that were described as memories, i.e. the slot table and the DMA Table are implemented as register arrays. The SPM memory is not part of the NI, and it will be implemented as a separate and independent memory block in the final integration with the processor(D3.8). The asynchronous routers require some careful handling of the timing for the synthesis procedure. Local timing constraints were applied in order to optimize the combinational logic and place delay elements of specific values. The routers were synthesized aiming for speed.

For the delay elements used in the asynchronous designs a trial synthesis and simulation step was done to find the appropriate delay value needed in each case. A delay element matching the HPU combinational delay and one matching the crossbar combinational delay are needed in the asynchronous routers. Additionally, delay elements are needed in the links, i.e. between routers and between NIs and routers. After determining the combinational delay, a safety margin of 20% was added to cover for delay fluctuations. The delay elements were implemented as a series of buffers and inverters. This was done by directing the synthesis tool to assign specific delay or delay ranges on a path. An optimization process for adjusting the delay elements, performing synthesis and simulation was repeated in order to achieve correct timing behavior (timing closure) of the asynchronous circuit without excessive delays compromising the performance.

To evaluate the MOUSETRAP router we synthesized and produced the layout. Table 1 shows the results for frequency, area and power consumption post-synthesis and post-layout for a number of asynchronous routers and a mesochronous one implemented in [5]. To further compare it with the

Table 1: Results for the router designs implementations.

	Cell Area	Post-synthesis		Post-layout	
		Freq.	Energy / cycle	Freq.	Energy / cycle
	um^2	MHz	pJ	MHz	pJ
Mesochronous	24239	1111	20.00	724	26.27
FIFOs	16213	1111	14.92	724	10.69
4ph-bd	7401	833	7.91	701	8.20
2ph-bd	7594	998	7.92	711	8.03
2ph-bd-g	7536	900		593	
Void 100 %			1.64		2.11
Void 30 %			6.51		7.23
Void 0 %			8.77		9.66
2ph-bd/LEDR-g	12578	862		645	
Void 100 %			3.82		3.80
Void 30 %			8.31		9.95
Void 0 %			10.02		12.00
mousetrap-g	7715	1126		746	
Void 100 %			1.28		2.17
Void 30 %			6.45		6.54
Void 0 %			8.24		8.39

routers in the Table 1 we also did a layout. After synthesis a gate-netlist of a MOUSETRAP router covers $7715\,um^2$ area that is much smaller than the mesochronous router. In terms of frequency it can be seen in the Table that our proposed router is faster than both the mesochronous and the other asynchronous routers. Finally, the energy per cycle consumed in this router is smaller than all the other designs, with the possibility of additional saving energy in case of idle traffic.

The synthesized NI in 65nm covers $12632\,um^2$, an as seen in [9] in 90nm covers $24000\,um^2$. This is significantly smaller than other existing NI designs. The maximum frequency of the synthesized NI can reach $1\,GHz$. This doesn't consider the timing of the SPM memory block.

For this deliverable a 2x2 NOC was synthesized. That includes 4 NIs, 4 routers and some connecting logic. After synthesis the total area of the gate-netlist of the NOC reaches $81388\,um^2$. The frequency of the NOC is the one that can be reached by the NIs, since the routers can reach a faster frequency, thus, $1\,GHz$.

5 Programming and Testing

To verify the design as well as the gate-netlist implementation, a 2x2 NOC was simulated with various test cases. In all cases both the VHDL model and the gate-netlist were tested. A test environment was created for this purpose. The test environment simulates the SPM memories and the processor interfaces.

The resetting and initialization of the NOC is done through the processor interface. The NI offers to the processor a 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 (for setting up connections and configuring transmission of packets). Each DMA table entry is mapped into a sequence of words in the global address space, as it is presented in Deliverable D3.2 [3].

As mentioned before, scheduling must be done on the level of phits. To obtain schedules on phit-level, a dedicated phit scheduler was used. The phit-scheduler can be found in [2]. The test environment is using the new phit-level scheduling. It produces the required signals for the OCP write commands to initialize the schedule into the slot table of each NI. At the same time OCP commands are produced to configure the routes in the DMA Tables. The configuration of DMA transfers can be done by OCP write commands to the DMA address space for the source address and destination address as well as the size of the message of the transfer and the enabling of the DMA operation.

To test the gate-netlist various test cases were simulated. Three of the test cases are included in the deliverable and follow the same core communication graphs as the test cases presented in [3]. The specific schedules were adapted to phit-level scheduling. Testcase0 is a simple test case. Testcase1 corresponds to one-to-all communication while testcase2 corresponds to all-to-all communication. With those 3 test cases, all the communication patterns in a 2x2 network are covered.

6 Source Access

The NOC code and related scripts are provided through the T-CREST repository via the `git` source code management tool:

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

The self-timed NOC source code along with the synthesis scripts, the gate netlist and the simulation environment can be found under the directory `async_noc`. The gate netlist along with delay information of the cells can be found under the directory `async_noc/netlists`. `async_noc/sim` directory contains build and simulation files for the simulation model and the gate netlist with configuration files for the test cases. Directory `async_noc/src` contains the VHDL source files. Finally, under directory `async_noc/synthesis` the scripts for synthesis are placed. They are to be used as a reference for the parameters of the design.

6.1 Requirements

To compile and simulate of the T-CREST NOC, the VHDL design as well as the gate netlist, 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 sufficient) [1]

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

To synthesize the gate netlist of the T-CREST NOC the following tools are needed:

- A Unix like environment with `git`, and a C/C++ compiler, such as: Linux, Mac OSX, or cygwin/Windows.
- DesignCompiler from Synopsys.
- STMicroelectronics Technology Libraries.

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
```

under the `async_noc/sim` directory will: (1) compile the VHDL design or the gate netlist, (2) simulate the model/netlist 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 as presented in Deliverable 3.2 [3] for the VHDL description model as well as for the gate netlist and clean up from temporary files:

`make test0` simulate testcase0

`make test1` simulate testcase1

`make test2` simulate testcase2

`make netlist0` simulate testcase0 for the gate netlist

`make netlist1` simulate testcase1 for the gate netlist

`make netlist2` simulate testcase2 for the gate netlist

`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`.

The synthesis of the T-CREST NOC is done using DesignCompiler from Synopsys and STMicroelectronics Technology Libraries. The process requires the run of the scripts in the DesignCompiler environment.

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 hardware implementation is time-predictable.

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

The hardware implementation provides mesochronous timing in the NIs and asynchronous timing in routers, thus supports GALS style design.

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

The hardware implementation provides communication channels between processing nodes. Communication to main memory is part of WP4.

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 supported in the hardware implementation.

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 hardware implementation supports pushing data from processing nodes to SPM in remote nodes. The transfer of data to the memory controller is part of WP4.

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

This is part of WP4.

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

This would imply excessive hardware cost thus, not supported.

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

The hardware implementation 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. Flow control is provided to the end user through access to status bits.

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

The hardware implementation 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 supported through DMA controllers.

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

This would imply excessive hardware cost thus, not supported.

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

It is part of WP4.

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

It is part of WP4.

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

It is part of WP4.

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

It is part of WP4.

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

A performance counter can be based on the NI clock, that is common throughout the system. But it is yet to be agreed with the industry partners.

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 hardware implementation 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).

The NOC hardware implementation 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

This deliverable report documents the hardware implementation of the self-timed NOC. This deliverable focuses on the design of processor-to-processor communication interconnect, as it is the most

demanding in terms of communication and analyzability. The self-timed NOC uses TDM scheme and supports mesochronous timing in the NIs, while having an asynchronous routers network.

The implementation of the self-timed NOC includes a parameterized design and a 2x2 gate netlist implementation. In this report the NI and the asynchronous router design were presented. The asynchronous router design presented in this report is a efficient design that emerged after the implementation and comparison of a number of asynchronous routers. It is based on the MOUSETRAP controller and applies a gating technique that saves considerable power in the case of idle traffic. It was shown to be faster, area and power efficient compared to other corresponding designs. The NI design is based on the new area-efficient design as it was described in the simulation model, in Deliverable 3.2, and in this report it is presented with the required changes to fit the network. It is shown that is more area-efficient than any others solution in the same area.

The implementation details as well as results from evaluating the network were shown in this deliverable. A 65nm technology library was used and the scripts for synthesis and simulation are included in the deliverable. The results give a characterization of the network and show the efficiency of the design. The implemented NOC was verified through simulation of test cases. The design along with the synthesis scripts and technology libraries can be used to implement various instances of the NOC of different sizes to be used in the T-CREST platform.

References

[1] Open core protocol - international partnership association. `http://www.ocpip.org`.

[2] Phit-level scheduler source code. `https://github.com/t-crest/SNTs`.

[3] T-crest project: D3.2 - simulation model of the self-timed noc. `http://www.t-crest.org/page/results`.

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

[5] E. Kasapaki, J. Sparsø, R.B. Sørensen, and K. Goossens. Router designs for an asynchronous time-division-multiplexed network-on-chip. In *Proceedings of the 16th Euromicro Conference on Digital System Design*, DSD '13, pages 319–326, 2013.

[6] Martin Schoeberl, Pascal Schleuniger, Wolfgang Puffitsch, Florian Brandner, Christian W. Probst, Sven Karlsson, and Tommy 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)*, pages 11–20, Grenoble, France, March 2011.

[7] M. Singh and S.M. Nowick. Mousetrap: ultra-high-speed transition-signaling asynchronous pipelines. In *Computer Design, 2001. ICCD 2001. Proceedings. 2001 International Conference on*, pages 9–17, 2001.

[8] J. Sparsø. Asynchronous circuit design – a tutorial. In J. Sparsø and S. Furber, editors, *Principles of asynchronous circuit design – A systems perspective*, chapter 1-8, pages 1–152. Kluwer Academic Publishers, 2001.

[9] Jens Sparsø, Evangelia Kasapaki, and Martin Schoeberl. An area-efficient network interface for a tdm-based network-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1044–1047, San Jose, CA, USA, 2013. EDA Consortium.