T-CREST

TIME-PREDICTABLE MULTI-CORE ARCHITECTURE
FOR EMBEDDED SYSTEMS

Project Number 288008

D 3.8 Integration report of the full system implemented on FPGA

Version 1.0
24 September 2014
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 José Neves Av. D. João II, Torre Fernão de Magalhães, 7 1998-025 Lisbon, Portugal Tel: +351 21 382 9366 E-mail: jose.neves@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

1 Introduction **2**

2 The Message-Passing NOC **3**

2.1 The Argo NOC – Architecture and Design 3

2.1.1 Architecture Overview 4

2.1.2 The Router 5

2.1.3 The Network Interface (NI) 6

2.2 Hardware Implementation and Evaluation 8

2.2.1 ASIC Implementations 8

2.2.2 FPGA Implementation 10

2.3 Timing Organization and Elasticity of the NOC 10

2.3.1 Timing Organization 10

2.3.2 Elasticity Analysis 10

3 Software support for the NOC **12**

3.1 The Scheduler 13

3.2 The Message Passing Library 13

3.3 The Aegean Configuration Tool 14

4 Integrated open-source platform for the Altera DE2-115 FPGA board **15**

4.1 Building the Platform 15

4.2 Results 16

5 Integrated Platform for the Xilinx ML605 FPGA Board **17**

6 Requirements **17**

7 Conclusion **19**

Document control

Version	Status	Date
0.1	First draft	27 August 2014
0.5	Second draft	18 September 2014
1.0	Final	24 September 2014

Executive summary

This document describes the deliverable *D 3.8 Integration report of the full system implemented on FPGA* of work package 3 of the T-CREST project, due 36 months after project start as stated in the Description of Work.

The design, performance, implementation, and FPGA-prototyping of the message-passing NOC developed in WP3 have been addressed in previous deliverables D 3.1 – D 3.7. The current document, D 3.8, concludes WP3 and gives a summary of the outcome. This includes: (i) a review of the design and implementation of the NOC developed in WP3, (ii) a description of the software tools developed in tasks 7.1 and 7.2 to support the integration and use of the NOC, and (iii) the final integration of the NOC into the complete T-CREST multi-core platform that is prototyped on an FPGA. Finally the report assesses the fulfillment of the requirements stated in D 1.1 “Evaluation Requirements”.

1 Introduction

This document *D 3.8 Integration report of the full system implemented on FPGA* is the final deliverable in work package 3 (WP3) of the T-CREST project.

The original Document of Work anticipated that WP3 would use an existing Network Interface unit developed at the Technical University of Eindhoven prior to the T-CREST project, and it anticipated that the T-CREST project would contribute a new asynchronous router design and an FPGA-implementation of the whole NOC. During the first year of the project, ideas for a novel and more efficient network interface architecture emerged, and as explained in D 9.2 "1st Annual Project Report" it was decided to pursue this and implement the new network interface. The extra effort required to do this was justified by an overall simpler design and a corresponding reduced integration effort. Another issue, which is more of a clarification, is that the network providing access to the shared memory has been covered in WP4 because of its tight integration with the memory controller.

As a result, WP3 has focused entirely on the message passing NOC, and WP3 has contributed a complete and innovative NOC-design comprising of asynchronous routers, a mesochronous timing organization, and a novel network interface architecture. The asynchronous router is roughly 2-3 times smaller than a corresponding clocked mesochronous router, and the new network interface architecture reduces the size of the network interface by a factor of 2-3. In addition to the deliverables in work package 3, the work has resulted in a number of publications: [17, 24, 26, 12, 23, 11, 15, 10, 14].

The eight deliverables of the work package are listed below. Previous deliverables D 3.1 through D 3.7 [2, 3, 4, 5, 6, 7, 8] have covered the design, performance and FPGA-implementation of the message-passing NOC.

D 3.1	Survey of time-predictable and asynchronous NOCs, and their WCET analysis
D 3.2	Simulation model of the self-timed NOC
D 3.3	Hardware implementation of the self-timed NOC
D 3.4	Report documenting hardware implementation of the NOC
D 3.5	Report on impact of asynchronicity on predictability of the NOC
D 3.6	FPGA implementation of self-timed NOC
D 3.7	Analysis report on FPGA implementation of self-timed NOC
D 3.8	Integration report of the full system implemented on FPGA

The current document, D 3.8, summarizes and concludes work package 3. This includes: (i) a review of the NOC-design and implementation developed in WP3, (ii) a description of the software tools developed in tasks 7.1 and 7.2 to support the integration and use of the NOC, and (iii) the final integration of the NOC in the complete T-CREST multi-core platform prototyped on an FPGA. Finally the report addresses the fulfillment of the requirements stated in D 1.1 Evaluation Requirements.

The T-CREST platform uses the memory controller developed in WP4. This memory controller is a proprietary design contributed by the Technical University of Eindhoven. As all other hardware components are open source, the Technical University of Denmark decided to implement a very simple memory controller, and arbiter [20] such that a fully open-source version of the T-CREST platform is also available.

Finally we note that the T-CREST platform is developed with an ASIC-implementation in mind and that it is being prototyped in an FPGA. For this reason, as discussed in D 3.5, we have implemented both a synchronous and an asynchronous version of the NOC-router.

As a result the NOC developed in WP3 has been integrated in the T-CREST platform and exhaustively tested in a number of scenarios including:

1. A synthesized 65 nm CMOS standard-cell implementation using asynchronous routers and verified in simulation. This constitutes the verification of the ASIC design.
2. A version of the open-source platform using asynchronous routers implemented and tested targeting the Xilinx ML605 FPGA board. Using this platform we verified the asynchronous NOC-design in a real running platform and we demonstrated its ability to tolerate skew between processor nodes.
3. A version of the open-source platform using synchronous routers, implemented and tested on an Altera DE2-115 FPGA board. In the early stages of the integration phase this platform was used by the Industry partners.
4. The final T-CREST platform, implemented and tested on the T-CREST Xilinx ML605 FPGA board. This platform uses the proprietary memory controller and the bluetree memory-NOC developed in WP4.

Item 1 is described in deliverables D 3.2, D 3.3, and D 3.4, and item 2 is described in deliverables D 3.6, and D 3.7. In this report we focus on items 3 and 4 – the full FPGA platform – and on the software tools developed to help users configure and program the platform. As D 3.8 is the final deliverable in WP3 we also include a brief summary of the complete NOC design.

The document is organized as follows. In Section 2 we present a brief overview of the NOC covering its architecture, design, implementation and evaluation. In Section 3 we present the software tools developed to support the integration process, i.e., tools for configuring and programming the platforms. In Section 4 we present the open-source platform (item 3 above). In Section 5 we present the final T-CREST platform (item 4 above). In Section 6 we assess the fulfillment of the requirements specified in D 1.1, and in Section 7 we conclude this report.

2 The Message-Passing NOC

In this section we provide a brief review of the NOC – called Argo – that has been developed in work package 3. The purpose is to make the report self-contained and the reader is referred to previous deliverables D 3.1 through D 3.7 and publications [26, 12, 11] for more details.

2.1 The Argo NOC – Architecture and Design

Argo is a statically scheduled time-division-multiplexed (TDM) NOC that implement virtual-circuits (point-to-point channels) over a source-routed packet-switched network of routers and links. The key

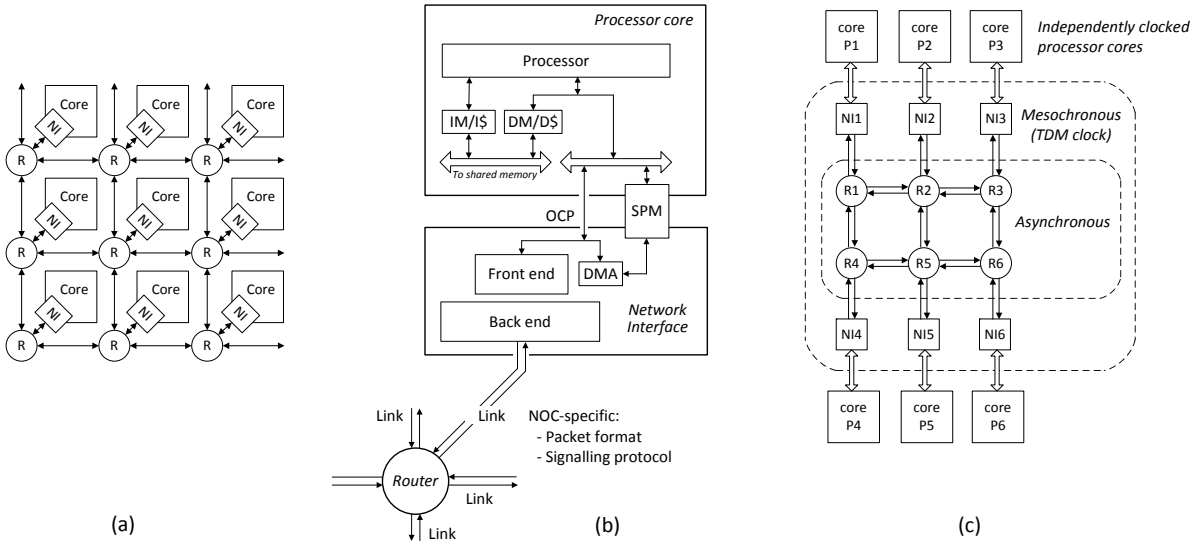

Figure 1: (a) Organization of the T-CREST multicore platform, (b) Block diagram of the T-CREST Network Interface, (c) Timing organization of the T-CREST platform.

features of Argo is a new network interface design and the use of asynchronous routers to provide time-elasticity as required by the mesochronous globally-asynchronous locally-synchronous (GALS) timing organization.

2.1.1 Architecture Overview

The architecture of the NOC appears in Figure 1(a). The routers are connected in a bi-torus topology forming a packet-switched network and each router is connected to a Patmos processor [21, 19] through a Network Interface (NI). The processors contain a number of caches and local scratchpad memories (SPM). The NOC implements DMA-driven block writes from the scratchpad-memory (SPM) of one processor to the SPM of another.

The effective design of the Argo NOC relies on two basic features that allow data to follow a direct path from SPM to SPM without any buffering, flow control or explicit synchronization.

The first feature is a novel NI design that integrates the DMA controllers with the TDM schedule and uses the dual-ported SPM for clock-domain-crossing. The block diagram of the NI design appears in Figure 1(b). In a TDM scheme the communication channels are assigned time-slots in the TDM schedule, such that each channel has some bandwidth guarantees. In the Argo architecture the DMA controllers are placed in the NIs and each DMA controller is responsible for the data transfer across one virtual channel. Controlled by the TDM-schedule the DMA controllers are activated one at a time, at the assigned time-slot and the result is a design that removes the need for arbitration, flow control and buffering in the NIs. In addition, the dual-ported SPMs can be used for clock domain crossing between the processors and the NIs. This removes the need for explicit clock-domain crossings for data between processors and NIs and buffering in the NIs.

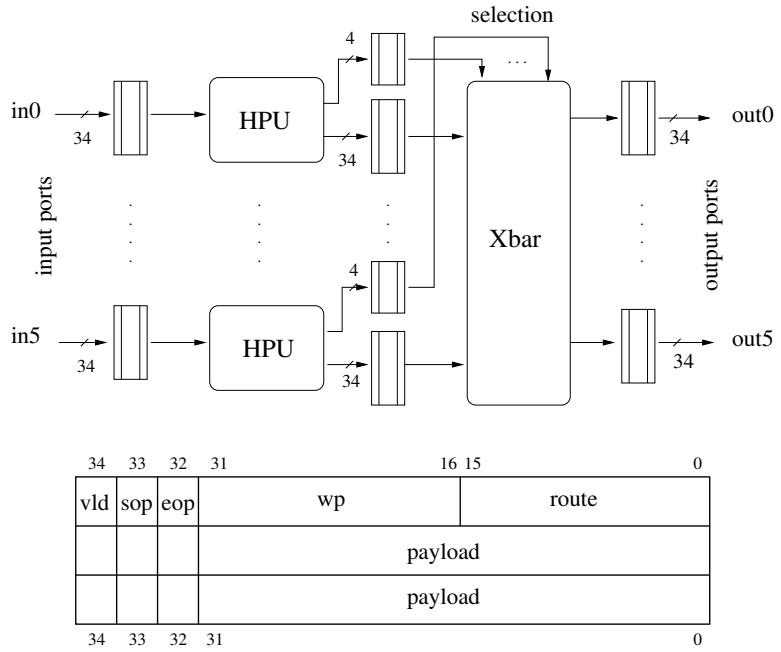

Figure 2: Block diagram showing the micro-architecture of the Argo asynchronous router and the packet format.

The second feature is the use of asynchronous routers. Asynchronous pipelines inherently behave as time-elastic ripple-FIFOs, and this removes the need for explicit synchronization between routers. The result is an asynchronous router that has smaller area and consumes less energy than a corresponding mesochronous clocked router

The timing organization of the NOC appears in Figure 1(c). It is a GALS organization using asynchronous routers and mesochronous NIs and it supports the use of individually clocked processors.

2.1.2 The Router

The TDM scheme, requiring no flow control or buffering in the routers, leads to a very simple router implementation. The Argo router is a 5-ported router with 3 pipeline stages: (1) link traversal, (2) header parsing unit (HPU), and (3) traversal of the crossbar switch (Xbar). The router pipeline appears in Figure 2.

Figure 2 also shows the packet format used in Argo. We use source-routing and the packet consists of one flit, thus the terms packet and flit denote the same. Each packet/flit consists of three phits: one header phit and two payload phits. Each phit is a 32-bit data word along with three control bits, indicating whether the phit is valid or not (vld), the start of packet (sop), and the end of packet (eop) respectively. The header phit contains the route (route) the packet follows, and the destination write address (wp).

To satisfy the needs for testing and use in the T-CREST platform, the Argo router has two implementations; a synchronous and an asynchronous. In the synchronous implementation the pipeline stages

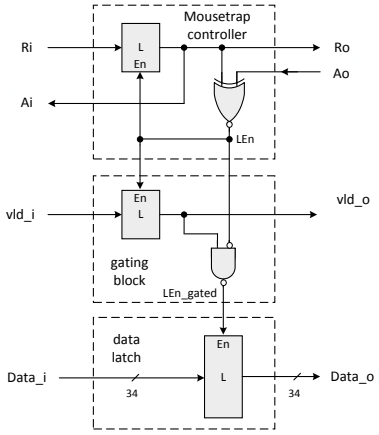

Figure 3: T-CREST handshake latch, consisting of three blocks: (i) the MOUSETRAP controller, (ii) the gating block and (iii) the normal enable data latch.

consist of clocked registers, and in the asynchronous implementation the pipeline stages consist of handshake latches.

The handshake latch of the asynchronous Argo router consists of a regular enable latch, a Mousetrap latch controller [22] and a "clock-gating" mechanism to save energy when the router propagate void phits (phits where the valid bit is not asserted). A schematic of the Argo handshake latch appears in Figure 3. The Mousetrap latch controller [22] implements a 2-phase (non return-to-zero) bundled-data handshake protocol. This implies delay elements in the request lines to match the propagation delays in the combinational logic. We chose the Mousetrap controller for its efficiency, after studying a range of alternative designs [12]. The "clock-gating" mechanism in the asynchronous handshake latch disables the enable signal to the data-latch when void phits are propagated.

2.1.3 The Network Interface (NI)

Figure 1(b) shows the block diagram of the NI and Figure 4 shows the micro-architecture of the NI. The key elements of the micro-architecture are the slot counter, the slot table, the DMA table, and the SPM. The slot counter is reset and it is incremented in all NIs using the same (mesochronous) clock. The slot counter defines the current slot in the TDM 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 an outgoing channel is assigned to this 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) as well as the route that the packet is to follow through the network of routers. When a DMA is active, the data is read from the local SPM from the read pointer address. At the destination the received data is written straight into the SPM at the write pointer address. The SPMs are dual-ported and a NI reads and writes to the SPM using its own clock. The NI-design is published in [26].

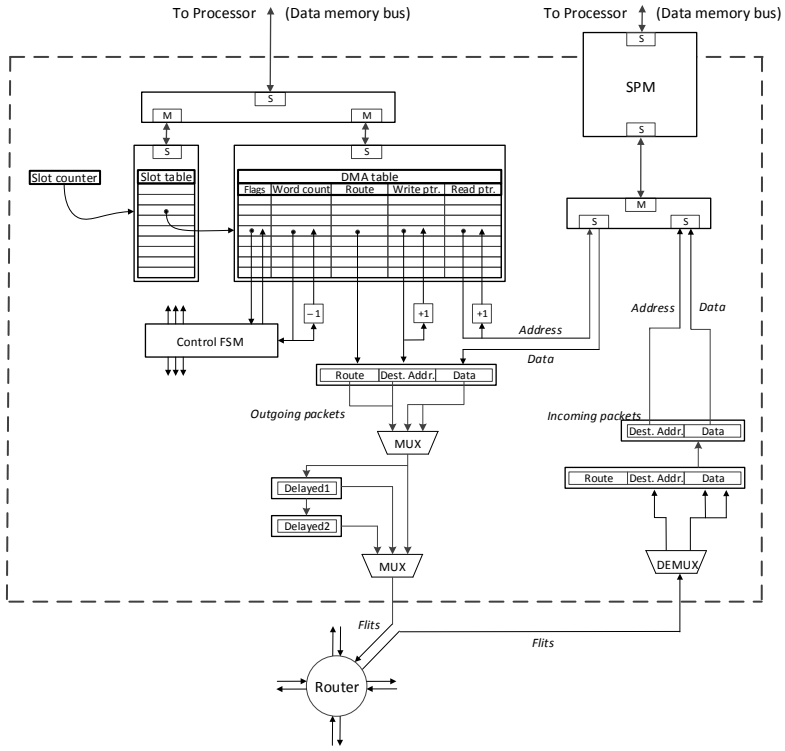

Figure 4: Block diagram showing the micro-architecture of the new NI for the inter processor communication NOC.

The duration of a TDM-slot is 3 clock cycles, matching the number of phits in a packet/flit. Since Argo uses phit-level scheduling, the NI must be able to start transmitting a packet/flit with a phase shift (delay) of 0, 1 or 2 cycles relative to the 3-cycle TDM-slots. For this reason the design includes two buffer registers, “Delayed1” and “Delayed2” as seen in Figure 4. The phase shifting of an outgoing packet/flit is controlled by a phase indicator that is part of the entry in the slot table.

It is assumed that the handshake cycle-time of the network of the asynchronous routers is shorter than the period of the NI-clock. This allows the NIs to send and receive phits at a rate controlled by the NI-clock. In this way there is no need for special synchronization circuitry between a router and an NI and the asynchronous handshake signals (request at the send side and acknowledge at the receive side) can simply be generated from the clock. [11] provides a detailed description of the interface between an NI and an asynchronous router and a detailed analysis of the amount of skew between the NIs that can be absorbed by the asynchronous routers.

The NI has two interfaces towards the processor side. One of the interfaces is an OCP [1] interface that is used to configure the NI, to set up DMA transfers, and to get status information from the DMA controllers. If the processor core and the NI lie in different clock domains, a clock-domain crossing is required on this interface. The second interface is for the data payload and is towards the SPM. As the SPM is a dual-ported memory it also provides clock-domain crossing of the payload data transferred between the NI and the processor.

Table 1: Results for alternative router 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
sync. router	8026				
FIFOs [16]	16213				
FIFOs [27]	(8106)				
Argo	7715	1126		746	
link util. 0 %			1.28		2.17
link util. 70 %			6.45		6.54
link util. 100 %			8.24		8.39

2.2 Hardware Implementation and Evaluation

The components of the NOC were implemented and evaluated for both ASIC and FPGA technologies. Several instances of the NOC were implemented using the already tested components in the two technologies, and in two versions; synchronous and asynchronous. The NOC was integrated with the Patmos cores and tested for both technologies. The functionality of the asynchronous version was additionally tested under skew.

2.2.1 ASIC Implementations

The asynchronous Argo NOC targets an ASIC implementation. Thus, we implemented and evaluated the individual components of the NOC as well as the entire NOC in ASIC technology. We used conventional EDA tools, and a 65 nm STMicroelectronics CMOS technology library. We used Synopsys Design Compiler for synthesis, ModelSim for simulation, SOC Encounter for layout, and Synopsys Prime Time for power analysis.

Router Implementations

The Argo router was implemented in two versions; a clocked one and an asynchronous one. Implementation results comparing different asynchronous versions of the router to a clocked mesochronous version were published in [12] and reported in [5]. Table 1 compares the asynchronous Argo router to its corresponding mesochronous implementation, for post-synthesis and post-layout results. As seen from the table, the speed and energy consumption of the two implementations are comparable with considerable energy savings of the asynchronous router on idle traffic. The area of the asynchronous router is significantly smaller than the mesochronous because it avoids the additional FIFOs for synchronization. The exact area reduction depends on how the FIFOs are implemented, but overall the area of the asynchronous Argo router is 2-3 times smaller than the area of the mesochronous router.

Table 2: Area figures for the NI ASIC implementations.

Instance Configuration			cell area
NOC dimension	#DMA ctrls	#time-slots	(um^2)
2x2	4	8	12632
4x4	16	23	30395

Table 3: Post Place & Route area and energy results for the 4x4 Argo NOC.

	Breakdown				Total
	Routers	NIs	Links	FIFOs	4x4 NOC
Cell area					
Total (um^2)	127446	537397	43289	12613	720745
Per node (um^2)	7965	33587	2706	788	45047
Relative (%)	17.68	74.56	6.01	1.75	100
Energy/cycle					
Total (pJ)	96.40	430.80	363.20	13.41	903.81
Per node (pJ)	6.03	26.93	22.70	0.84	56.49
Relative (%)	10.67	47.66	40.19	1.48	100

Network Interface Implementations

Table 2 shows area results for two instances of the NI with different configurations. The difference in size in the larger instance comes from the storage requirements for the increased number of DMA controllers (4 and 16 per node respectively) and the larger schedule. An instance of the NI with 4 DMA controllers and 8 timeslots requires 12632 um^2, as seen in Table 2. This is significantly smaller than other existing NI designs. As explained in [26] the NI is 2-3 times smaller than previously published NIs for real-time TDM-based NOCs.

A 16-Node NOC Implementation

To test the entire design of the asynchronous Argo NOC, a 2x2 and a 4x4 instance of the NOC were implemented. Table 3 presents the post place and route results on area and energy per cycle. The 4x4 instance includes 16 NIs, 16 asynchronous routers, and local FIFOs between NIs and routers. These local input and output FIFOs are one and two stages deep respectively, and they have been added for symmetry reasons. The area breakdown shows that 75 % is attributed to the NIs and 18 % of the area is attributed to the routers both of which are already 2-3 times smaller than previous designs, demonstrating the area efficiency of the NOC. We can also observe that routers are energy efficient (11 % of the total energy consumption), while the links consume a significant amount of energy (40 %).

Table 4: Resource usage on the ML605 FPGA board.

	Self-timed NOC				Synchronous
	2x2 NOC	NI	IO FIFO	Router	Router
Number of Slice-Registers	5,440	672	108	580	545
used as Flip Flops	2,688	672	0	0	545
used as Latches	2,752	0	108	580	0
Number of Slice LUTs	3,985	438	19	538	423
used as logic	3,985	438	19	538	423
used as Memory	0	0	0	0	0

2.2.2 FPGA Implementation

The prototyping platform is the Xilinx ML605 FPGA board. Therefore, the NOC was implemented and tested in FPGA technology. The implementation details are reported in Deliverables D 3.6 [7] and D 3.7 [8]. Table 4 presents the resource usage of the NOC and its elements on the ML605 FPGA board. The correct operation of NOC was verified in the presence of skew between the NIs.

2.3 Timing Organization and Elasticity of the NOC

The timing behavior of the asynchronous NOC is explored in [11] and it is presented in Deliverables D3.5 [6] and D3.7 [8]. This section serves as an overview.

2.3.1 Timing Organization

The T-CREST platform supports a GALS timing organization. The timing organization appears in Figure 1(c). The processors can be individually clocked, the NIs are mesochronously clocked and they form a shell around the core network of asynchronous routers. The NIs drive the network of asynchronous routers, by producing and consuming phits, thereby enforcing a data-rate that corresponds to the NI-clock.

Any asynchronous pipeline inherently operates as an elastic FIFO. In this way the elastic behavior of the network of asynchronous routers connected to clocked NIs is similar to the elastic behavior of a structure of asynchronous FIFOs connected to clocked producers and consumers at its endpoints.

The TDM router, as presented in the Deliverable Reports D3.2 [3] and D3.4 [5] consists of 3 pipeline stages for each of the five ports. The crossbar implements a join-fork (JF) functionality and is a point of synchronization for all five pipelines. With this understanding, the network of routers can be seen as a structure of elastic FIFOs connected by join-fork-points. Figure 5 illustrates this view for a 2x3 NOC.

2.3.2 Elasticity Analysis

The elastic structure described above, can absorb some amount of skew. Skew between the NIs will cause the FIFO-segments to be more filled or drained than the ideal state established at reset. Such

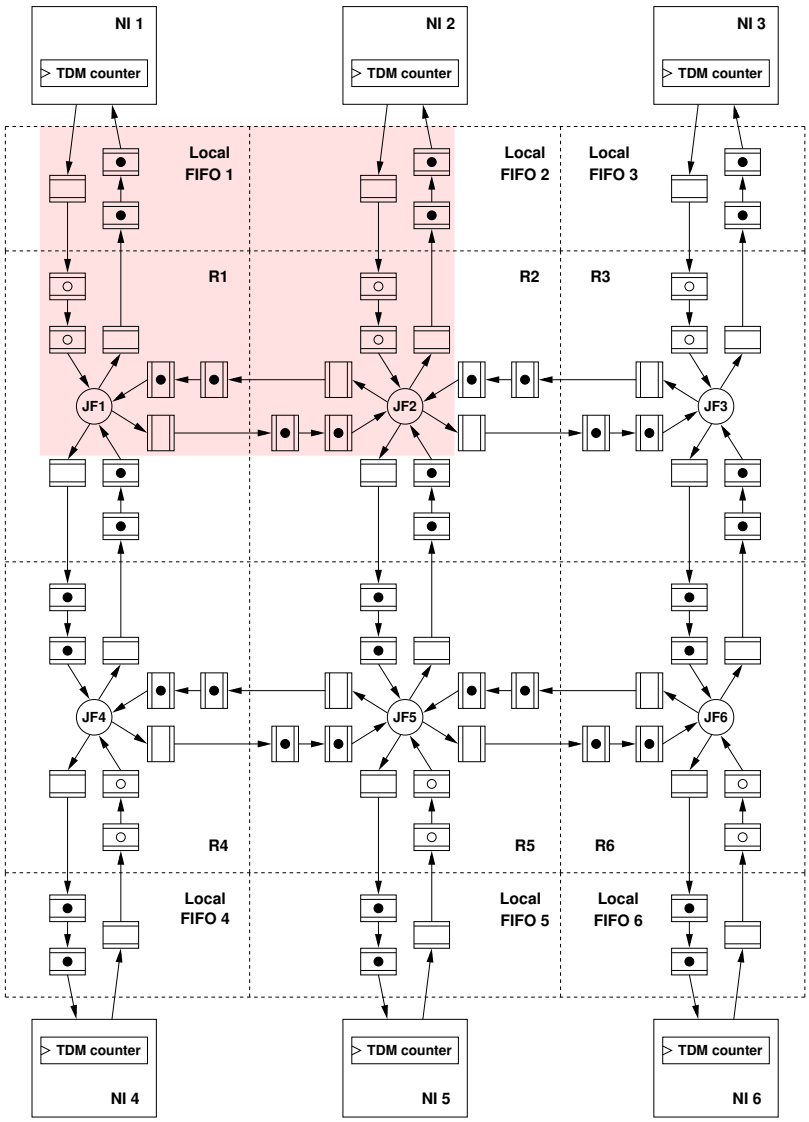

Figure 5: FIFOs and timing organization.

filling or draining reduces the speed of a FIFO-segment [25, Section 4.3]. As explained earlier the entire design is based on the assumption that the asynchronous network of routers is faster than the clocked NIs. When the skew increases beyond some value this assumption no longer holds, and it is interesting to explore how much skew can be tolerated before the assumption breaks.

To evaluate this we performed a timing analysis on a model of the structure of the NOC, Figure 5, and smaller parts of it. We used a Signal Transition Graph (STG) model and performed a Timing Separation of Events (TSE) [9] analysis. For this purpose we used a TSE analysis tool [13]. The analysis provides a mathematical worst-case bound on the maximum timing separation between events, taking into consideration the initial behavior of the circuit from reset and until a repetitive steady state of operation.

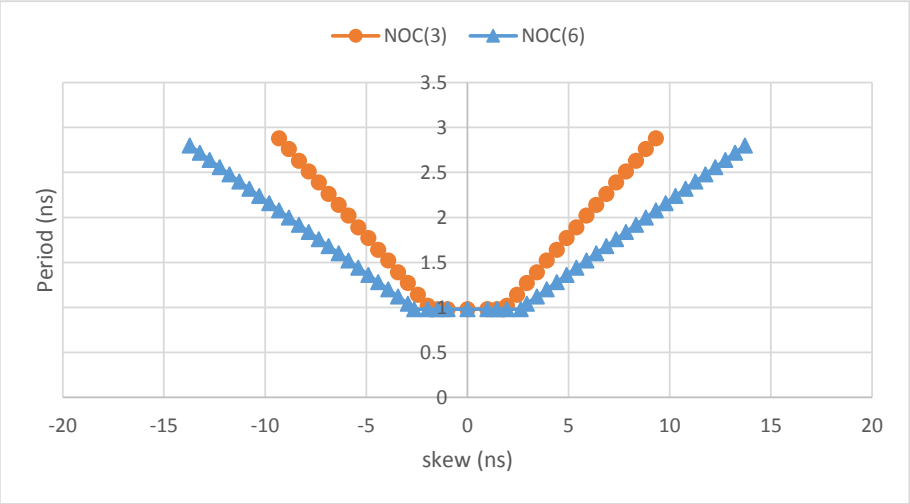

Figure 6: Skew bounds.

Using this TSE analysis tool, for a certain NI-clock period, we can verify the correct operation of the NOC for different skew values. In this way we can determine a relation between the NI-clock period and the maximum amount of skew that can be tolerated. The analysis uses typical and constant delay values derived from a 65 nm CMOS post-synthesis standard cell implementation of a complete NOC.

Results of the analysis are shown in Figure 6. The figure shows two curves referring to a NOC with three and with six stages of input/output FIFOs. The results show that at the maximum frequency (1.0 GHz) the skew between NIs can reach 1.8 ns (1.8 cycles) with three-stage input/output FIFOs and 2.63 ns (2.9 cycles) with six-stage input/output FIFOs. At $500\,MHz$ the maximum absorbed skew increases to $5.8\,ns$ (2.9 cycles) and $9.4\,ns$ (4.7 cycles) respectively.

This skew tolerance is surprisingly large and even at the maximum speed of 1.0 GHz it exceeds the expected skew among NIs in a real implementation. The detailed analysis and more results can be found in [11] and in Deliverable D 3.7 [8].

3 Software support for the NOC

This section describes the software tools developed under Tasks 7.1 and 7.2 to support the integration process of the Argo NOC with the Patmos processor cores. The software tools are used to configure the NOC and to program the integrated platform. The software support includes the following tools:

Poseidon a scheduler generating the static schedule for the communication over the NOC,

MP library a message passing library used by the application programmers to pass messages over the NOC from one task to another,

Aegean a configuration tool to automatically generate an instance of the multi-core platform comprising the Argo NOC and Patmos processors.

3.1 The Scheduler

To generate a static schedule that implements a set of virtual channels with on top of the Argo NOC, a scheduler was developed. The scheduler, presented in [23], produces a communication schedule at the level of phits, such that there are no collisions in the NOC and such that the requirements for communication bandwidth and latency are met.

For each application that is executed on a multi-processor system, the workload is distributed into tasks which then are mapped to a set of processors for execution. Depending on the mapping of tasks to processors, certain pairs of processors have certain communication bandwidth requirements. The communication requirements are defined as required bandwidth, in MB/s, between pairs of processors in the platform. The bandwidth requirements between all pairs of processors are the overall communication requirement that needs to be scheduled.

The scheduler takes two parameters to generate a schedule. The first parameter is a core communication graph describing the bandwidth requirements between all pairs of processors in the platform. The second parameter is a platform graph describing the topology of the platform and any additional pipelining inserted for performance reasons.

In the asynchronous router the three pipeline stages can store up to two phits. This is different from the packet length of three phits (header, payload1, payload2), and as a result phits from different packets following different routes may become unaligned when they arrive at a router. For this reason the scheduling must be done at the level of phits rather than at the coarser level of flits/packets.

The scheduler always creates a schedule that fulfills the normalized bandwidth graph. The schedule has to be repeated at a frequency that provide the required absolute bandwidths. This may exceed the maximum speed of the NIs. This means that the application should be restructured or that a larger or faster platform should be used.

3.2 The Message Passing Library

To support the integrated platform and the application programmers using the NOC, a message passing library was developed. The library is written in C code and offers the application programmer a toolset to send and receive messages through the NOC.

The message passing (MP) library offers the application programmer the following functions:

`mp_send_init` Function for initializing the state of the send function.

`mp_recv_init` Function for initializing the state of the receive function.

`mp_nbsend` Non-blocking function for passing a message to a remote processor under flow control. The data to be passed by the function should be in the local message buffer in the SPM before the function is called. Checks remote buffer for availability and returns if remote buffer is not available or if remote buffer is ready and the DMA controller has been setup.

`mp_send` Blocking function for passing a message to a remote processor under flow control. The data to be passed by the function should be in the local message buffer in the SPM before the function is called. Checks remote buffer for availability and returns when remote buffer is ready and after the DMA controller has been setup.

`mp_nbrecv` Non-blocking function for receiving a message from a remote processor under flow control. The data that is received is placed in a message buffer in the SPM. When the received message is no longer used, the reception of the message should be acknowledged with the function `mp_ack`. Returns when message is received or if there is no message.

`mp_recv` Blocking function for receiving a message from a remote processor under flow control. The data that is received is placed in a message buffer in the SPM. When the received message is no longer used the reception of the message should be acknowledged with the function `mp_ack`. Returns when the message is received.

`mp_nback` Non-blocking function for acknowledging the reception of a message. This function should be used with extra care, if no acknowledgment is sent the communication channel will be blocked until an acknowledgment is sent. This function shall be called to release space in the receiving buffer when the received data is no longer used.

`mp_ack` Function for acknowledging the reception of a message. This function shall be called to release space in the receiving buffer when the received data is no longer used.

`mp_send_alloc_size` Function for returning the amount of data that the channel is allocating in the sending SPM.

`mp_recv_alloc_size` Function for returning the amount of data that the channel is allocating in the receiving SPM.

3.3 The Aegean Configuration Tool

A configuration tool, named Aegean, was developed to support the integration process. The tool is a collection of Python scripts that automatically generate an instance of the open-source multi-core platform under a specified configuration. This platform includes a number of Patmos processors connected by the message-passing Argo NOC, and an on-chip memory and memory controller, targeting an Altera DE2-115 FPGA board. The Aegean configuration tool was developed to help the integration and testing of the NOC with Patmos processors in the early stages of integration and does not include the proprietary memory-controller and the memory NOC developed by WP4.

The configuration of a given platform is described in XML files. The parameters that can be defined in the hardware platform configuration are the following:

- the number of Patmos cores, i.e. the dimensions of the platform,
- the topology and connectivity of nodes in the Argo NOC,
- specific configuration of Patmos processors,
- configuration of IO devices,
- configuration of the target FPGA board,
- main memory size,
- the communication requirements of the application, i.e. all-to-all communication.

According to the configuration provided, the Aegean tool generates the hardware description of the platform defined and the communication schedule for the NOC. First, the communication requirements are used to generate a schedule for the NOC. Second, the hardware description of the different

components of the platform are generated. Third, the top levels in the hierarchy are generated, the components are instantiated and connected. Finally, the Aegean tool-suit offers automatic compilation and programming of the platform on the FPGA board.

4 Integrated open-source platform for the Altera DE2-115 FPGA board

This platform refers to item 3 as presented in the Introduction. It is a fully integrated platform including a number of Patmos cores, connected by the Argo NOC, on-chip memory and memory controller, and IO devices. The platform is automatically generated by the Aegean tool and the dimensions of the platform, i.e. number of cores, size of memory, is a configuration parameter, as described in Section 3.3. This platform was developed and used in the early stages of integration to help in the integration and testing process of the Argo NOC with the Patmos cores. The platform is open-source and it does not include the proprietary memory controller of WP4.

The instances of this platform were implemented and tested on an Altera DE2-115 FPGA board.

4.1 Building the Platform

In the following we present the build instructions for the platform using the Aegean configuration tool on a Linux/Ubuntu system. The default platform is a 2x2 platform, including 4 Patmos processors connected by the Argo NOC and a memory arbiter giving access to the shared memory.

The platform can be checked out from GitHub as part of the T-CREST project. The T-CREST project can be found in `$HOME/t-crest` and the open source platform along with the Aegean configuration tool can be checked out and build with the following commands:

```
mkdir ~/t-crest
cd ~/t-crest
git clone https://github.com/t-crest/patmos-misc.git misc
./misc/build.sh
```

Several packages and tools need to be installed and set-up. The list of packages along with detailed setup instructions can be found in [18].

The whole build process of the 2x2 platform, in C, configuration of the FPGA, and downloading an application is `Makefile` based. The build of the platform is done in the aegean folder, therefore, the following descriptions assumes you have changed to:

```
t-crest/aegean
```

The platform is generated by the command:

Table 5: Resource utilization of the open-source integrated platform on the Altera DE2-115 FPGA board.

		Combinational Cells	Register Cells	Memory Bits
2x2 platform		44410	25557	582912
1 node	SPM memory	12	4	32768
	Patmos proc	9343	4763	113920
	Argo NI	562	667	0
	Argo router	716	565	0

```
make platform
```

To synthesize the platform for the FPGA board, run:

```
make synth
```

The synthesized platform is configured on the FPGA with the command:

```
make config
```

The default board is the Altera DE2-115 FPGA board. The target board can be changed by setting the `BOARD` variable in the above make commands. The currently supported boards are:

- Altera DE2-70 (`altde2-70`)
- Altera DE2-115 (`altde2-115`)
- Altera/Farnell BeMicro (`bemicro`)

For a different platform configuration the corresponding .xml file should be written and placed under the `t-crest/aegean/config` directory.

4.2 Results

The resource utilization of the open-source 2x2 platform implemented for the Altera DE2-115 FPGA board is shown in Table 5. The table shows the Cells used for combination logic in the third column, the Cells used for registers in the fourth column and the number of memory bits in the fourth column. The numbers in the table show the area efficiency of the NOC. It can be seen that for a single node of the platform 88 % of the Combinational Cells of the node are used for the Patmos processor, while 5 % are used for the NI and 7 % for the router. These numbers will scale linearly with the number of nodes in the platform.

Table 6: Total resource utilization of the final T-CREST platform implemented on the Xilinx ML605 FPGA board.

Slice Logic Utilization	
Number of Slice Registers	32960
Number of Slice LUTs	38829
Number used as Logic	36814
Number used as Memory	2015
IO Utilization	
Number of IOs	83
Specific Feature Utilization	
Number of Block RAM/FIFO	115
Number using Block RAM only	115
Number of BUFG/BUFGCTRLs	6
Number of DSP48E1s	32

5 Integrated Platform for the Xilinx ML605 FPGA Board

The final T-CREST platform that is used for evaluation by the industry partners corresponds to item 4 mentioned in Section 1. The platform comprises a number of Patmos processors developed by WP2, connected by the Argo NOC developed by WP3. Access to shared memory is provided through the memory tree NOC developed by WP4 and is controlled by the memory controller developed by WP4. The platform is implemented and runs on the Xilinx ML605 FPGA board.

Each component of the platform was developed by the individual work packages and the top level hierarchy instantiating and connecting the components was developed manually for different dimensions of the platform. Two instances of the platfrom have been provided to the industry partners, a 2x2 and a 4x4 instance. The architecture of the platform is able to support up to a 8x8 platform. However the size of Xilinx ML605 FPGA board does not allow platforms larger than 4x4 to be implemented.

The resource utilization for the final T-CREST integrated platform, in the 2x2 version, appears in Table 6. The numbers are total numbers for the entire platform and they are comparable to the open-source platform presented in Section 4.

6 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 single clock of the NIs is driving the asynchronous network enforcing a static schedule according to TDM scheme. Since the schedule is static and timing is based on a single NI-clock, the NOC is time-predictable.

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

The NoC provides mesochronous timing in the NIs and asynchronous timing in routers, thus supporting a GALS style design.

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

The NoC provides communication channels between processing nodes. Communication with 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 by the NoC.

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 NoC supports pushing data from the SPM of processing cores to SPM in remote cores. 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.

Implementing this requires additional hardware and substantially longer schedules. It has been decided not to support this in hardware. Pulling of data can be supported in software if needed.

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

The NoC is configurable at initialization. Processors set up the slot table and routes in the NIs.

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

The NOC itself does not need flow control; due to the TDM-scheme it is non-blocking. At the user level the message passing library offers primitives that implement flow control in software.

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

The NoC supports up to 64 tiles. The only factor limiting to 64 tiles is the packet format. This can easily be extended.

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 and longer schedules (similar to N-3-048). It has been decided not to support this in hardware. It can be supported in software if needed.

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

This is part of WP4.

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

This is part of WP4.

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

This is part of WP4.

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

This is part of WP4.

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

There is already a performance counter in the processing cores which can be used for the NOC as well. In case a performance counter of the NOC is needed, it can be implemented by extending the slot counter in the NIs.

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 static schedule produced for the NoC does not allow 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 contributes bounded latency as argued in D 3.5.

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 done in bounded time. All actions that refer to the DRAM controller are satisfied by the memory NOC developed by WP4.

7 Conclusion

This document is the final Deliverable report, D 3.8, for work package 3. It summarized the contribution of WP3 and described the final integrated platform of the T-CREST project.

The main contribution of WP3 is Argo, a novel and efficient NOC design with an elastic timing behavior. This document presented an overview of the design and its implementation. The results show that the Network Interface (NI) is 2-3 time smaller than alternative NI designs and that the

asynchronous router to be 2-3 times smaller than corresponding clocked mesochronous designs. In addition, the elastic behavior of the NOC was explored and it was showed that the NOC can absorb 1-3 cycles of skew depending on the operating frequency, while preserving the correct operation.

This report also presented the software that was developed to support Argo and the integration process. Software support for the use of the Argo includes the Poseidon scheduler to generate the static TDM communication schedule, and a message passing library to provide a message passing API to the application programmer. Software to support integration includes the Aegean configuration tool for the automatic generation of specific instances of the open-source integrated platform.

Finally, this report presented two integrated platforms using the Argo NOC design. One is the final integrated T-CREST platform implemented on the Xilinx ML605 FPGA board – the platform that is used for the final evaluation by the industry partners. The other is an alternative fully open-source platform implemented on an Altera DE2-115 FPGA board. This platform can be generated automatically by the Aegean configuration tool.

Finally the report reviewed the requirements stated in D 1.1 “Evaluation Requirements” and commented on their level of fulfillment.

References

[1] Open COre Protocol - International Partnership Association. http://www.ocpip.org.

[2] T-crest project: D3.1 - Survey of time-predictable and asynchronous NOCs, and their WCET analysis. http://www.t-crest.org/page/results.

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

[4] T-crest project: D3.3 - Hardware implementation of the self-timed NOC. http://www.t-crest.org/page/results.

[5] T-crest project: D3.4 - Report documenting the hardware implementation of the self-timed NOC. http://www.t-crest.org/page/results.

[6] T-crest project: D3.5 - Report on impact of asynchronicity on predictability of the NOC. http://www.t-crest.org/page/results.

[7] T-crest project: D3.6 - FPGA implementation of self-timed NOC. http://www.t-crest.org/page/results.

[8] T-crest project: D3.7 - Analysis report on FPGA implementation of self-timed NOC. http://www.t-crest.org/page/results.

[9] H. Hulgaard, S.M. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. *Computers, IEEE Transactions on*, 44(11):1306–1317, 1995.

[10] E. Kasapaki, M. Schoeberl, Rasmus Bo Sørensen, C. T. Müller, K. Goossens, and J. Sparsø. Argo: A Real-Time Network-on-Chip Architecture with an Efficient GALS Implementation. In *IEEE Transactions on VLSI Systems*, 2014. submitted.

[11] E. Kasapaki and J. Sparsø. Argo: A Time-Elastic Time-Division-Multiplexed NOC Using Asynchronous Routers. In *Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 45–52, May 2014.

[12] E. Kasapaki, J. Sparsø, R.B. Sørensen, and K. Goossens. Router Designs for an Asynchronous Time-Division-Multiplexed Network-on-Chip. In *Proc. of Euromicro Conference on Digital System Design (DSD)*, pages 319–326, Sept 2013.

[13] Evangelia Kasapaki. An EDA tool for the timing analysis, optimization and timing validation of asynchronous circuits. Master's thesis, Computer Science Department, University of Crete, Greece, Heraklion, Crete, Greece, April 2008.

[14] I Kotleas, D.R. Humphreys, R.B. Sørensen, E. Kasapaki, F. Brandnery, and J. Sparsø. A Loosely Synchronizing Asynchronous Router for TDM-Scheduled NOCs. In *Proc. IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 151–158, 2014.

[15] C. T. Müller, E. Kasapaki, Rasmus Bo Sørensen, and J. Sparsø. Synthesis and Layout of an Asynchronous Network-on-Chip using standard EDA Tools. In *Proc. of Nordic Microelectronics event (NORCHIP)*, 2014. to appear.

[16] I. Miro Panades and A. Greiner. Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in gals architectures. In *Proc. IEEE/ACM Intl. Symposium on Networks-on-Chip (NOCS)*, pages 83–92, 2007.

[17] 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, May 2012.

[18] Martin Schoeberl, Florian Brandner, Stefan Hepp, Wolfgang Puffitsch, and Daniel Prokesch. Patmos reference handbook.

[19] Martin Schoeberl, Florian Brandner, Stefan Hepp, Wolfgang Puffitsch, and Daniel Prokesch. Patmos reference handbook. Technical report, Technical University of Denmark, 2014.

[20] Martin Schoeberl, David VH Chong, Wolfgang Puffitsch, and Jens Sparsø. A time-predictable memory network-on-chip. In *Proceedings of the 14th International Workshop on Worst-Case Execution Time Analysis (WCET 2014)*, 2014.

[21] 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.

[22] M. Singh and SM Nowick. MOUSETRAP: Ultra-high-speed transition-signaling asynchronous pipelines. In *Intl. Conference on Computer Design (ICCD)*, pages 9–17. IEEE Computer Society Press, 2001.

[23] Rasmus Bo Sørensen, Jens Sparsø, Mark Ruvald Pedersen, and Jaspur Højgaard. A metaheuristic scheduler for time division multiplexed network-on-chip. In *Software Technologies for Future Embedded and Ubiquitous Systems (SEUS), 2014*. IEEE, 2014.

[24] J. Sparsø. Networks-on-chip for real-time multi-processor systems-on-chip. In *Proc. International Conference on Application of Concurrency to System Design (ACSD)*, pages 1–5, June 2012.

[25] J. Sparsø and S. Furber, editors. *Principles of asynchronous circuit design – A systems perspective*. Kluwer Academic Publishers, 2001.

[26] J. Sparsø, E. Kasapaki, and M. Schoeberl. An area-efficient network interface for a TDM-based Network-on-Chip. In *Proc. Design Automation and Test in Europe (DATE)*, pages 1044–1047, March 2013.

[27] P. Wielage, J.E. Marinissen, M. Altheimer, and C. Wouters. Design and DfT of a high-speed area-efficient embedded asynchronous FIFO. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 853–858, 2007.