T-CREST

TIME-PREDICTABLE MULTI-CORE ARCHITECTURE
FOR EMBEDDED SYSTEMS

Project Number 288008

D 3.7 Analysis report on FPGA implementation of self-timed NOC

Version 1.0
30 June 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

Document control

Version	Status	Date
0.1	First draft	25 May 2014
0.2	Second draft	1 June 2014
0.3	Third draft	15 June 2014
1.0	FInal	30 June 2014

Executive summary

This document describes the deliverable *D 3.7 Analysis report on FPGA implementation of self-timed NOC* of work package 3 of the T-CREST project, due 33 months after project start as stated in the Description of Work.

The document presents a testcase application that is run on a complete 2x2 platform that uses the self-timed message passing NoC. The NoC is named Argo. The 2x2 platform is tested in a post place & route simulation and on a Xilinx ML605 FPGA board. The self-timed NOC and the entire platform work correctly and the experiments show that the asynchronous NOC is capable of absorbing considerable skew between the mesochronous Network Interfaces. The document describes how to synthesize and run the testcase.

Finally the report provides an analysis of the elasticity of the NoC, i.e., how much skew it can tolerate between the mesochronous network interfaces. The analysis comprises both analytical bounds on the maximum skew tolerance and an analysis of a detailed gate-level model of a network of routers.

1 Introduction

In this report we demonstrate a case study of the self-timed NoC integrated in the T-CREST platform using the Aegean tool-suite which was developed in Task 7.2 to support integration. We report the resources used for the FPGA implementation of the NoC and we explore its operation under skew. Finally, a theoretical analysis of the elasticity of the self-timed NoC is presented.

2 FPGA prototype of the self-timed NoC

The self-timed NoC was tested in a complete T-CREST platform. The platform consists of four Patmos processors, a shared memory structure implemented using the on-chip memory of the FPGA, and the self-timed NOC that supports the message passing among the four processors. The Aegean configuration tool-suite developed in Task 7.2 was used to configure and integrate all components into a complete instance of a 2x2 T-CREST platform. This platform has been successfully tested in a post-place & route simulation and as well as on the actual Xilinx ML605 FPGA board.

The NOC consists of four NIs and four routers as they were presented in Deliverable Reports D3.2 [1] and D3.4 [1, 2] and published in [7, 4]. The routers are asynchronous designs implementing the 2-phase bundled-data protocol for synchronization and a gating mechanism to reduce power consumption. Some small local FIFOs are used between the NIs and routers to provide some additional skew tolerance, as it will be explained in Section 5.

The application used is called "hello_sum_reduced" and it is synthesized with the platform and bootable from the FPGA memory. The master processor, with id 0, sends to the slave processor 1 the message "Hello slaves sum_id:0" through the MPI. Processor 1 receives the message, adds its id to the sum and sends the message to Processor 2. The same is repeated by Processor 2 to Processor 3 and finally Processor 3 sends the string with the total sum back to the master. The master processor utilizes the serial port to send out the results of the operation. The expected output is:

> MASTER: message sent: Hello slaves sum_id:0
>
> MASTER: finished polling
>
> MASTER: message received: Hello master sum_id:6

3 Accessing the code and building the platform

In this section the requirements and the instructions on how to build the T-CREST platform with the self-time NoC and the Aegean tool-suite are provided.

3.1 Accessing the code

The source files for the Argo NoC and the Aegean tool-suite are provided via the `git` source code management tool.

The code for Argo is located at: `https://github.com/t-crest/argo`.

The Aegean suite is located at: `https://github.com/t-crest/aegean`.

3.2 Requirements

To make, synthesize and simulate the T-CREST platform with the self-timed NoC for the Xilinx ML605 board, the following tools are needed:

Make platform

- A Unix like environment with `git` , such as: Linux, Mac OSX, or cygwin/Windows
- The T-CREST platform tool-chain installed

Synthesize

- Xilinx ISE

Post Place and Route Simulate

- ModelSim for simulation (full version)
- Xilinx CompXlib

3.3 Make platform

Before making the platform some parameters need to be specified to describe the platform.

Configuration xml file Under `aegean/config/` there are some configuration xml files. The one used for this case study is *ml605_oc_async_fpga.xml*. It specifies a 2x2 system with bi-torus NoC topology, router depth of 3 stages and link depth of 0 stages (links not pipelined) and self-timed NoC targeting an FPGA implementation. Then the intellectual properties (*ml605ocm* and *ml605ocs*) and their location on the NoC are specified. The shared memory is set to on-chip (*OCRam*) and the requested schedule to *all2all*. The name of the xml configuration file defines the name of the project. Make a copy of this file and name it after the project name of your selection (*project_name.xml*).

Makefile parameters Under `aegean/` create a file named config.mk. In this file it is specified which project is targeted. Additionally, to enable pre-synthesis simulations using Xilinx simulation primitives, the paths to the compiled libraries need to be provided.

AEGEAN_PLATFORM=project_name

SECUREIPPATH=/path_to_secureip_compiled_library

UNISIMPATH=/path_to_unisim_compiled_library

Set application Under `aegean/config/ip/` open the file *ip.xml*, locate the intellectual property *ml605ocm* and set *<bootrom app="bootable-hello_sum_reduced"/>*. Do the same for the intellectual property *ml605ocs*.

Make platform Open the terminal, navigate to the `aegean/` folder and run *make projectname* to verify what is the current project and *make platform* to make the project. As a result, a folder named after the project is generated under `aegean/build/` .

Pre-synthesis simulation After making the platform, run *make sim-fpga* under the `aegean/` folder to run a simulation of the platform targeting the FPGA implementation.

3.4 Synthesis and testing

The synthesis and layout of the NOC for the Xilinx ML605 board is done using Xilinx ISE, following the steps below:

1. Open Xilinx project *aegean/build/project_name/ise/project_name_async.xise* with Xilinx ISE.
2. Set the constraints *MAX_FANOUT* and *NO_BUFFER* when synthesizing the self-timed NOC in order to satisfy timing requirements, and set *KEEP_HIERARCHY* when synthesizing the platform in order to ease the debugging process.
3. If wanted, generate the post place & route simulation model by running *Implement Design -> Place & Route -> Generate Post-Place & Route Simulation Model* under the *Processes* pane.
4. Run *Generate Programming File* under *Processes* pane.

To run simulation after Place&Route the Xilinx library models need to be compiled for ModelSim. This is done using Xilinx CompXlib tool. The library model required is simprim_ver.

1. Copy the files:

   aegean/vhdl/sim/par_example/aegean_testbench_POST.vhd
   aegean/vhdl/sim/par_example/task.do
   aegean/vhdl/sim/par_example/waves.do

   in `aegean/build/project_name/`.

2. Modify task.do to point to the location of simprim_ver library and the glbl.v file that were created by CompXlib.

3. Open aegean/build/project_name/ise/netgen/par/aegean_top_timesim.v with your favorite text editor. Replace *SIM_COLLISION_CHECK('ALL')* with *SIM_COLLISION_CHECK('NONE')*. Then search for *tx_baud_tick*. Locate *tx_baud_tick_XXXX* where *XXXX* is a number.

4. In the testbench file aegean_testbench_POST.vhd replace *tx_baud_tick* with *tx_baud_tick_XXXX* as shown in the exported Verilog Post-Place&Route simulation model.

5. Open Modelsim, change directory to `aegean/build/project_name` and run *do task.do*.

4 FPGA implementation results

4.1 Resource usage on the Xilinx 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

Table 1: Resource usage on the ML605 FPGA board.

Table 4.1 shows the resource usage for the 2x2 Argo NOC when implemented on the Xiling ML605 FPGA board. The numbers have been extracted from a synthesis report for the whole platform synthesized with the *KEEP_HIERARCHY* attribute. The synchronous and the self-timed NOC use the same clocked NI; only the router is different.

It is seen that the asynchronous router is slightly bigger than the synchronous one. There are three reasons for this: (i) The matched delay-elements that are used in two-phase bundled-data asynchronous circuits, (ii) the use of a few extra latches in the header parsing control circuit and (iii) duplication of logic in order to avoid fan-out problems for gated latch-enable signals in the router datapath. The number of delay elements were trimmed (down) in a repeated post-place and route simulation.

4.2 Operation under skew

For reasons of prototyping, skew was inserted by delaying the reset of some network adapters by one clock cycle. The skew inserted with regard to the topology is shown in Figure 1. The skew enforced in NI counters in nodes (0,0) and (1,1) is $-\delta$ and the skew enforced in NI counters in nodes (0,1) and

(1,0) is $+\delta$ in respect to the clock signal. Figure 2 shows the clock signal and the values of the four slot counters, in nodes (0,0), (0,1), (1,0) and (1,1) for a situation where the skew is 1 cycle ($\delta = 0.5$ cycles). This demonstrates the timing elasticity of the NOC.

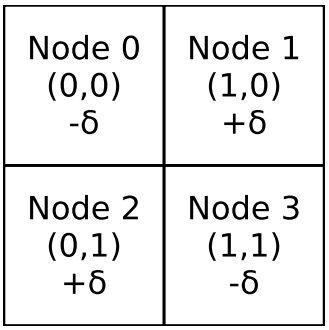

Figure 1: Skew allocation between NIs

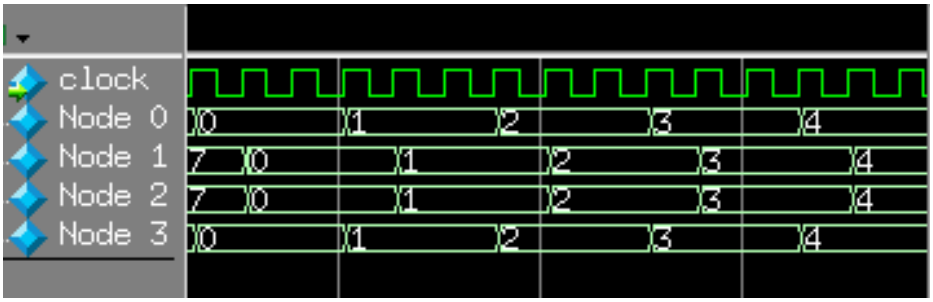

Figure 2: Modelsim post place and route simulation waveforms showing the clock signal and the slot counters in the NIs.

5 Analysis of the timing elasticity of the self-timed NoC

To evaluate the elastic behavior of the self-timed NOC, besides running a number of simulations, a theoretical analysis was done to explore the limits of its elasticity. This involves a theoretical performance analysis of the basic building blocks of the NOC as well as the definition of safe bounds for the overall NOC 5.2. Additionally, it includes a modeling of the NOC and an exploration of the maximum skew that the self-timed NOC model can tolerate in relation to the operating frequency 5.3. The elasticity analysis is presented here and a more detailed analysis and results are published in [6].

As will become clear, the skew tolerance is expressed relative to the clock period, and the skew tolerance is a dimensionless number that depends on structural properties of the circuit. For this reason we offer here the same analysis as performed in [6] that assumes gate delays from the 65 nm CMOS cell library used in deliverable D3.3.

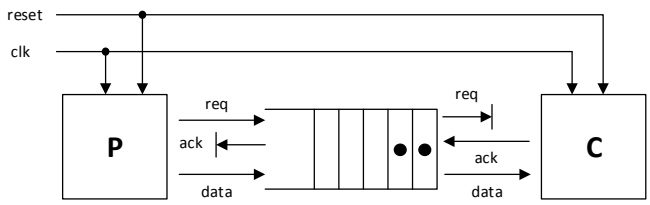

Figure 3: FIFO structure showing the basic concept of the elasticity of Argo design.

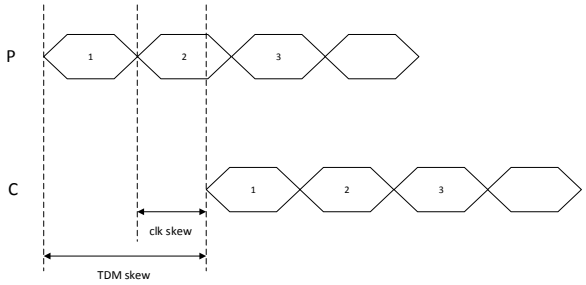

Figure 4: Timing diagram of data flow between producer and consumer, showing TDM skew.

5.1 Timing organization

Any asynchronous pipeline can be seen as an elastic FIFO, with every handshake latch stage of the asynchronous pipeline being a stage of the FIFO. In this way the elasticity of the structure of asynchronous routers connected to synchronous NIs is similar to the elasticity of a FIFOs structure connected to a clocked producer and consumer at its endpoints. The simplest building block of such a structure is one FIFO connected to a producer and a consumer as Figure 3 shows. A producer (P) and a consumer (C) connected at the endpoints of the FIFO can drive the data flow through the FIFO. A clock signal (clk) enforces the rate of data flow while signal reset sets the FIFO to its initial state.

In a TDM context, reset initializes the TDM counters in P and C and clk drives the TDM counters. Both clk and reset signals can present skew between P and C, filling or draining the FIFO. In a data flow sequence, as Figure 4 shows, skew on clk is the phase difference between clk pulses as observed between P and C clk while reset is a time difference in initialization. The total skew from clk and reset might exceed one clock period and in a TDM context it results in a difference in the TDM counters. This is the skew the NoC needs to absorb to operate correctly.

The TDM router, as presented in the Deliverable Reports D3.2 [1] and D3.4 [2], is a 5-ported router consisting of 3 stages of pipeline for each of the five ports. Crossbar implements a join-fork functionality and is a point of synchronization for all five pipelines. Considering each asynchronous pipeline stage as a FIFO stage, and the crossbar as a join-fork point (JF), a 2x3 NOC can be seen as a structure of FIFOs and synchronization points as Figure 5 illustrates. The FIFOs connecting two synchronization points can provide an amount of timing elasticity. The FIFOs connecting a clocked NI and a synchronization point provide an additional amount of elasticity. Thus, between any two NIs an amount of elasticity is present and an amount of skew can be absorbed. To estimate the maximum skew that can be absorbed, in the following analysis we consider smaller parts of the FIFO structures present in Figure 5.

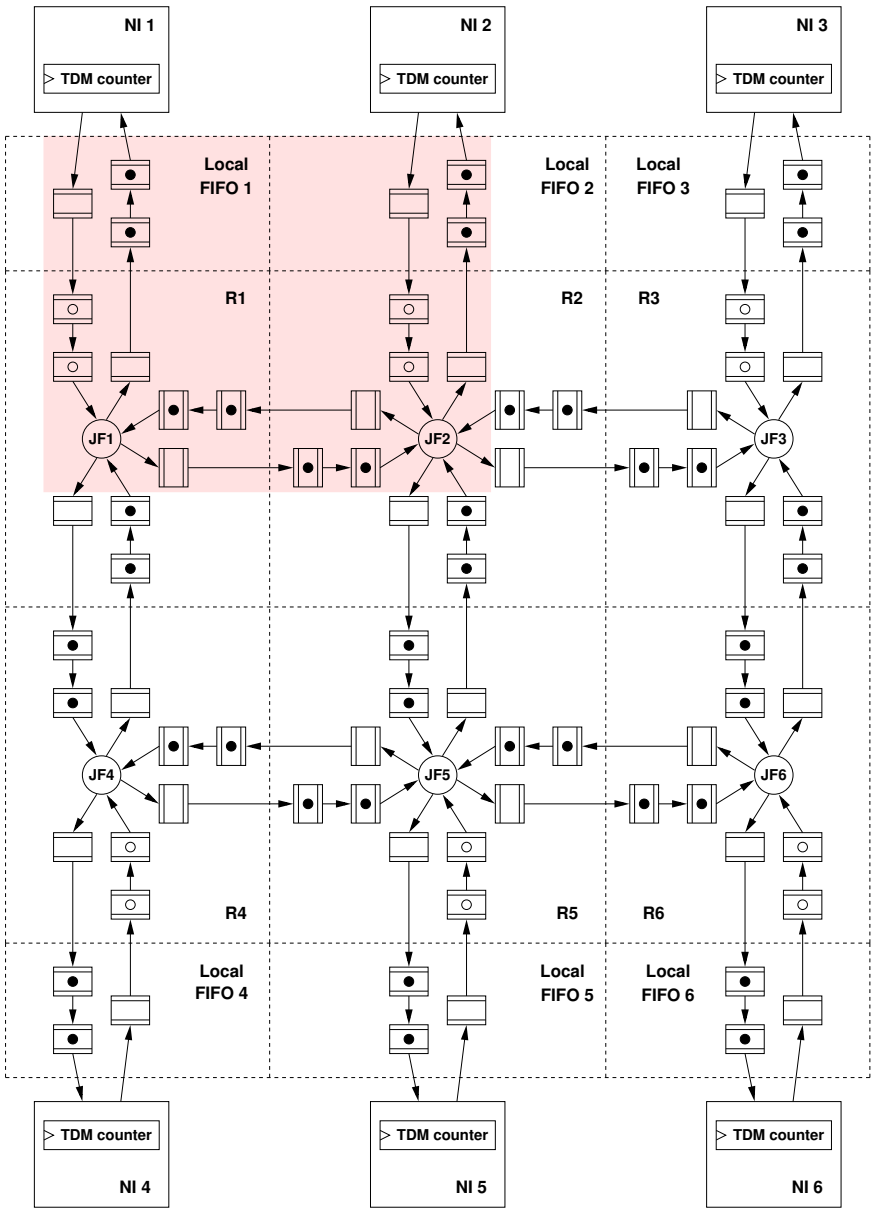

Figure 5: FIFOs and timing organization.

5.2 Elasticity analysis

To understand the consequences of skew throughout the NOC we consider the repeated structure of a FIFO ring connecting two JF points as it appears in Figure 6. The clock signal, clk, enters the join and determines (i.e. limits) the rate at which data-tokens flow in the structure. Producer, JF1, and consumer, JF2, operate at the same rate but possibly with some skew. If JF2 is lagging behind JF1 ($d1 = 0, d2 > 0$) then the FIFO will be more filled than in its initial state, and if the skew is in the other direction ($d1 > 0, d2 = 0$) it will be drained. Both situations may reduce the inherent speed of the FIFO. For the lower FIFO the situation is symmetric except for the swapped roles of JF1 and JF2.

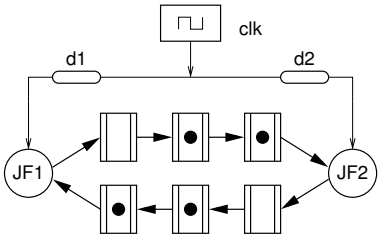

Figure 6: Two neighboring join-fork points connected by three-stage ripple FIFOs.

We denote the maximum skew between two join-fork nodes δ_{JF}. In the example in Figure 6 it is somewhat less than two clock cycles. Arguments similar to the above can be made for a circuit fragment consisting of an NI and the FIFOs that connect it to its nearest JF node. We denote the maximum skew between a NI and its nearest JF node δ_{NI}. In the following we analyze a subcircuit of the NOC, that, although small, it is representative of the entire design. The subcircuit is illustrated by the shaded area in Figure 5 and considering the two neighboring NIs of the shaded area, we see that the maximum possible skew between the two neighboring NIs is:

$$\delta \leq 2\,\delta_{NI} + \delta_{JF} \tag{1}$$

The maximum skew between NIs that are further apart could potentially be higher, as the path between them contains more JF to JF segments. But as we are interested in the worst-case, this is not relevant. Furthermore, the operation of a JF point is synchronized with its neighboring JF points, which in turn synchronize with their neighbors etc. For this reason the skew between neighboring JF points may never reach δ_{JF}. In the worst case it could be of the opposite direction and thus contribute negatively, resulting in the following worst case bound on the maximum skew between any pair of NIs in the NOC is:

$$\delta_{NOC} \leq 2\,\delta_{NI} - \delta_{JF} \tag{2}$$

This is a perhaps a too pessimistic bound but it is a safe bound.

5.3 Modeling and system analysis

The next step is to establish analytical values for δ_{NI} and δ_{JF} and ultimately δ_{NOC}. We do this using STG models of the parts of the circuit in the shaded area in Figure 5. These STGs model the structure of the latch controllers, joins, forks and matched delay elements. The analysis is based on the Mousetrap latch controller that we use.

A handshake latch consisting of a Mousetrap latch controller, a gating block and an enable latch in the datapath appears in Figure 7(a). The operation of a Mousetrap latch controller is modeled by the STG in Figure 7(b). Signals (Ri, Ai) and (Ro, Ao) represent the input and output handshake channels respectively. Signals $En+$ and $En-$ represent the enabling and disabling of the latch. The edges in the graph represent dependencies between signal events. For the timing analysis we associate component delays to the edges of the STGs as seen in Figure 7(b) where t_L is the propagation delay of the latch and t_X the delay of the XNOR gate. The dependency $Ai \rightarrow Ro$ is a zero-delay dependency, as Ai and Ro are the same signal, i.e. produced concurrently. Dotted edges represent timing dependencies

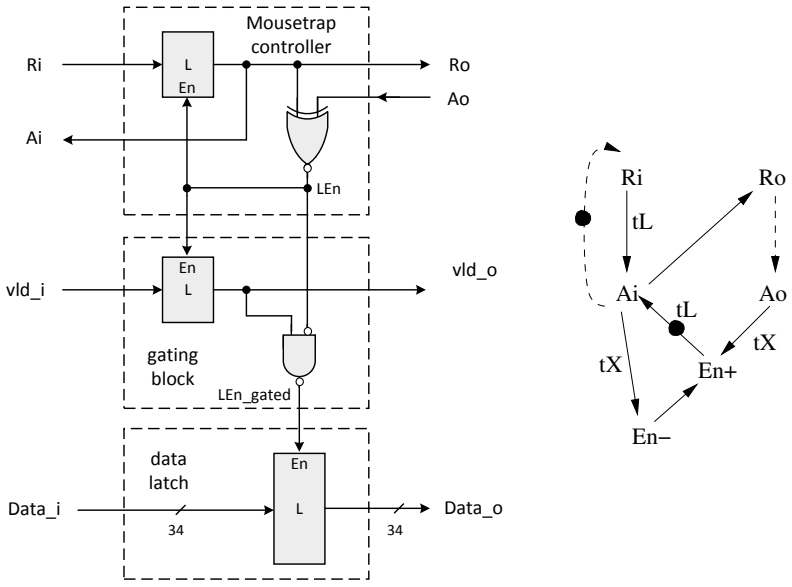

Figure 7: (a) A handshake latch consisting of a Mousetrap latch controller, a gating block and an enable latch in the datapath. (b) STG model of the Mousetrap latch controller.

that model the environment. STGs of more complicated structures can be derived by combining the STGs of the individual parts in a tile-based way.

To analyze the STG models, we used a TSE (Timing Separation of Events) analysis tool [5] that is based on the TSE algorithm of Hulgaard et al. [3]. The analysis provides a mathematical worst-case bound on the maximum timing separation between events, including the initial behavior of the circuit from reset to steady state of operation.

Using this TSE analysis tool we can check, for different skew values, that the network of asynchronous routers is still functioning correctly under the period of the NI clock, and by gradually increasing the skew we can determine the bound. The analysis uses typical and constant delay values derived from a 65 nm CMOS post-synthesis standard cell implementation of a complete NOC.

5.4 Analysis Results

Based on the described analysis we present results on the skew that a number of structures can absorb as obtained from the STG analysis. Figure 8 shows the relationship between the skew and the period of the NI clock for different circuit fragments: *NOC* denotes the end-to-end NI-to-NI path, representing δ_{NOC}. $RING$ denotes the ring segment connecting two JF points with three FIFO stages, representing δ_{JF}. $IO - FIFO$ denotes a segment connecting an NI and a JF point with three FIFO stages, representing δ_{NI}.

The graphs show that at the maximum frequency 1 GHz the skew between NIs can reach 1.8 ns (1.8 cycles). At $500\,MHz$ the maximum skew increases to $5.8\,ns$ (2.9 cycles). We can observe that the amount of absorbed skew, i.e. the elasticity of the self-timed NOC, is increasing with a lower

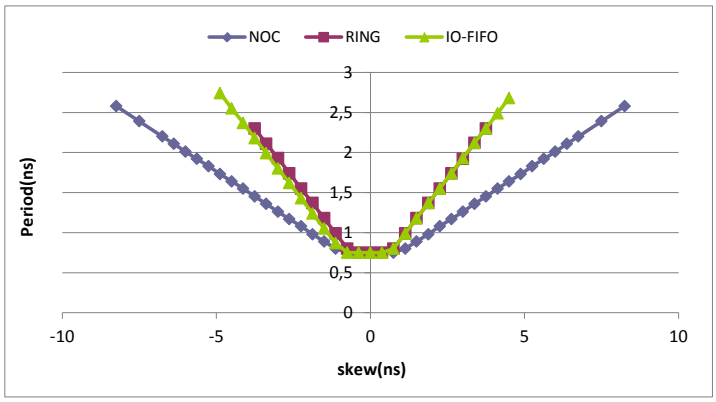

Figure 8: Skew analysis of a JF-JF segment, a NI-JF segment, and the complete end-to-end NI-NI path.

operating frequency, however, even at maximum frequency there is a considerable amount of skew absorbed between the two NIs.

6 Conclusion

This document presented an analysis on the FPGA implementation of the self-timed NOC. A testcase was used to show the correct functionality of the self-timed NOC on an FPGA and a model analysis was used to prove the elasticity bounds of the self-timed NOC under skew.

The current document described the implementation of a 2x2 instance of the self-timed NOC integrated with T-CREST platform on the Xilinx ML605 FPGA board. Instructions were given on how to build the 2x2 platform with the self-timed NOC and an analysis on FPGA resources was provided. The platform was proven to operate correctly under skew through simulation and running on the actual board.

An analysis on the elasticity of the NOC was also presented. A theoretical analysis provided bounds of skew within which the self-timed NOC can operate correctly and a model analysis confirmed these bounds, giving a relation of the skew to the operating frequency. It was shown that the self-timed NOC can absorb at least 1.8 cycles of skew when the NOC is operated at its maximum frequency. If the NI clock frequency is reduced – the normal mode of operation – the skew tolerance increases to 3 cycles or more.

References

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

[2] T-crest project: D3.4 - report documenting the hardware implementation of the self-timed noc. `http://www.t-crest.org/page/results`.

[3] 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.

[4] E. Kasapaki, J. Sparso, R.B. Sorensen, and K. Goossens. Router designs for an asynchronous time-division-multiplexed network-on-chip. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 319–326, Sept 2013.

[5] 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.

[6] Evangelia Kasapaki and Jens Sparso. Argo: A time-elastic time-division-multiplexed noc using asynchronous routers. In *Asynchronous Circuits and Systems (ASYNC), 2014 20th IEEE International Symposium on*, pages 45–52, May 2014.

[7] 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.