T-CREST

TIME-PREDICTABLE MULTI-CORE ARCHITECTURE
FOR EMBEDDED SYSTEMS

Project Number 288008

D 3.6 FPGA implementation of self-timed NOC

Version 1.0
31 March 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	18 March 2014
0.2	Second draft	19 March 2014
0.8	Draft for internal revision	20 March 2014
1.0	Final	31 March 2014

Executive Summary

This document describes the deliverable *D 3.6 FPGA implementation of self-timed NOC* of work package 3 of the T-CREST project, due 30 months after project start as stated in the Description of Work.

The document presents a FPGA implementation of the self-timed message-passing network-on-chip (NOC) that is used in the T-CREST platform. The NOC design is the same as presented in deliverables D3.2 - D3.4. This design consists of mesochronously clocked network interfaces (NIs) and asynchronous routers. This report deals with prototyping of the asynchronous NOC in FPGA technology, and more specifically the implementation of the special asynchronous components that are used in the router design. The FPGA implementation of the asynchronous router has been verified in a post synthesis, post post-place&route simulation of a small NOC, and is ready for integration with the remaining parts of the platform.

The source code and the FPGA implementation along with the corresponding scripts for synthesis, place&route 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 FPGA implementation. The functionality has been verified by running the all-to-all communication testcase as it was presented in Deliverables D3.2 - D3.4.

1 Introduction

Deliverable D3.6 is the FPGA implementation of the self-timed NOC. The NOC is a statically-scheduled time-division-multiplexed (TDM) NOC, and it has already been presented in deliverables D3.2 - D3.4 [1, 2, 3]. The design involves a mesochronously clocked network interfaces (NI) design, and asynchronous routers. The asynchronous routers are connected in a bi-torus topology and exhibit a time-elastic behavior that enables a globally asynchronous locally synchronous implementation of the NOC and of the entire T-CREST platform.

The current deliverable, D3.6, refers to the preparation of the self-timed NOC for the integration with the processors on the Xilinx ML605 FPGA-board. As FPGAs are designed and used for implementation of clocked synchronous circuits it is necessary to implement a number of basic asynchronous components in FPGA-technology. This report addresses the design and implementation of these components using the look-up tables (LUTs) offered by the FPGA technology. More specifically this involves the implementation of C-elements and matched delay-elements.

This report documents: (i) the design of these asynchronous components, and (ii) the resulting router implementation and its verification in post-synthesis and post place & route simulation of a 2x2 network.

2 Asynchronous router and FPGA adaptations

The asynchronous router utilizes: (i) Mousetrap controllers [4] to implement pipeline stages, (ii) matched delays to guarantee bundling constraints (needed to satisfy setup and hold time requirements) and (iii) asynchronous C-elements to synchronize the request and acknowledge signals from the five router ports.

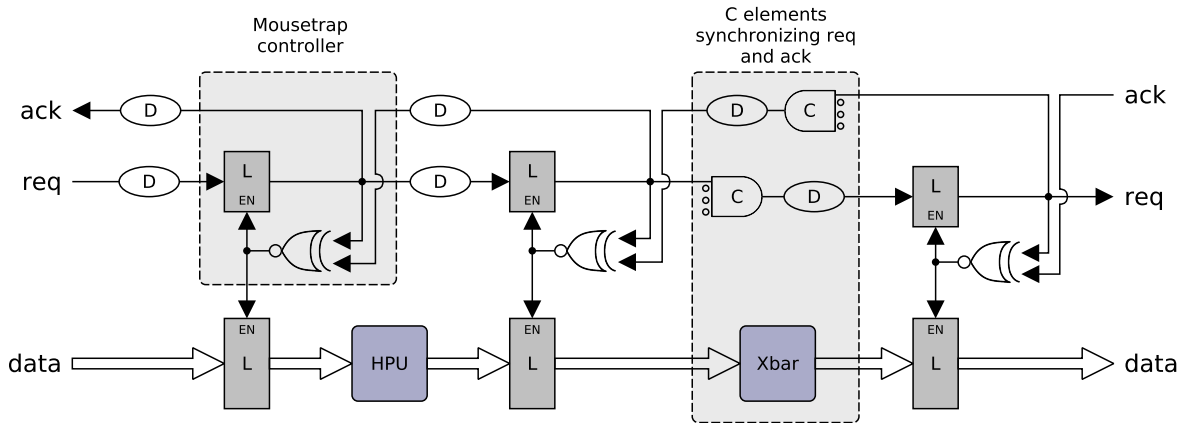


Figure 1: Router design

The C-elements are components with state. Their functionality is to wait for all inputs to go HIGH in order for their output to be set HIGH (like an AND gate) and subsequently wait for inputs to go LOW before setting the output to LOW (like an OR gate). Therefore, the output of a C-element is a function of both its current input and output. Synchronization of 2-phase handshake channels is

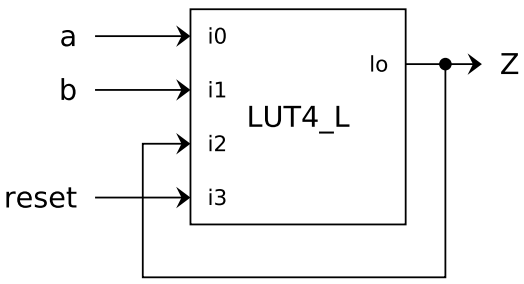
achieved with C-elements by waiting all of the port requests to trigger (either to HIGH or to LOW) and afterwards by waiting all of the ports to acknowledge the synchronized request.

As mentioned before and seen in Figure 1, matched delays are used in the design. Three reasons enforce the usage of delay elements: 1) combinatorial path delay that has to be matched so that the data is stable when the request triggers, 2) latch setup time requirements and 3) latch hold time requirements. The delays of type 1 & 2 are on the path of the request signals, but delay type 3 is on the backward acknowledgment path.

Special care needs to be taken when targeting an FPGA to implement the pre-mentioned asynchronous components.

3 C-Elements and Matched Delays on Xilinx FPGA

The C elements are components whose output is dependent both on the inputs and the current state of the component. A 4 input LUT is used to provide this functionality on FPGA by using the unisim library of Xilinx to instantiate a LUT4_L for every C-element. Two of the inputs of the LUT are used for the reset and the output feedback and the values of the LUT are hard-written with the appropriate vector (Figure 2).



reset i3	Z i2	a i1	b i0	Z lo
0	x	x	x	reset value
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

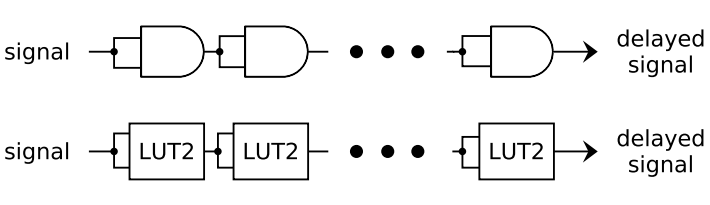
Figure 2: A C-Element LUT implementation

The VHDL code implementing a C element in LUT technology appears in Listing 1.

When introducing delay, no distinguishing is made between rising and falling edges of signals, the reason being the use of a 2-phase handshaking protocol. An array of LUTs is sufficient to implement such a delay, which grows with the length of the array. The logic function of each LUT is an AND gate, but since both inputs are connected to the same signal, each LUT behaves as a simple delay (Figure 3).

Listing 1: VHDL implementation of a C element for FPGA technology.

```
--Constant using the generic "reset_value"
constant rv : bit := reset_value;
constant reset_vector : bit_vector(7 downto 0) := rv&rv&rv&rv&rv&rv&rv&rv;

signal s_out : std_logic;

begin

C_element: LUT4_L
  generic map (INIT => "11101000"&reset_vector)
  port map (I0 => in1, I1 => in2, I2 => s_out, I3 => reset, LO => s_out);

--------------
-- Connect the outputs to the correct signals
--------------
  z <= s_out;

end architecture; --LUT
```



i1	i0	I0
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3: Delay element implementation as an array of LUTs.

4 Testing

To test the asynchronous components (i.e., C-elements and delay elements) in FPGA-technology, the asynchronous router was synthesized and its functionality was verified on the Xilinx ML605 FPGA-board using switches and LEDs as inputs and outputs.

In order to further test the self-timed NOC, a complete 2x2 bi-torus NOC was synthesized. The synthesized system includes the asynchronous routers and the synchronous NIs. This NOC was tested through post-synthesis simulation using the same testcase with all-to-all communication as in the Deliverables D3.2 - D3.4 [1, 2, 3].

Additional testing was done through post-place&route simulation. This simulation considers the actual circuit and wire delays after place & route; an important step to ensure correct functionality of an asynchronous circuit.

The placed&routed on FPGA design includes the self-timed part of the NOC, i.e. the asynchronous routers. The NIs do not require special timing handling and additionally, the IO pins of the Xilinx ML605 board are not enough to accommodate the NI ports. The latter is the reason that testing on FPGA can only be done after integration with Patmos processor. Thus, the post-place&route simulation considers a placed&routed 2x2 network of routers and a simulated environment of NIs and SPMs running the same testcase used for post-synthesis simulation as it was presented in the Deliverables D3.2 - D3.4 [1, 2, 3]. The correct post-place&route simulation ensures the correct functionality of the self-timed NOC and makes it ready to be integrated with the processors and the rest of the T-CREST platform.

The work presented in this report completes Task 3.4 and Deliverable 3.6 and makes the self-timed NOC ready to be integrated with processors, Task 3.5 and Deliverables 3.7 and 3.8.

5 Accessing the Source Code

The source and environment files are provided through the t-crest repository via the `git` source code management tool:

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

The additional code for the Xilinx FPGA targeted implementations, together with the simulation testbenches and the associated scripts can be found under the directory `/async_noc_FPGA`.

5.1 Requirements

To synthesize and simulate the T-CREST NOC for the Xilinx ML605 board, the following tools are needed:

- A Unix like environment with `git` , such as: Linux, Mac OSX, or cygwin/Windows
- ModelSim for simulation (full version)
- Xilinx ISE
- Xilinx CompXlib

5.2 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/argo.git
```

or downloaded as .zip file from GitHub:

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

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

- Create a project in ISE with name "xilinx" under /async_noc_FPGA/
- Load source files in project
- Synthesize
- Place & Route
- Generate Place & Route Simulation Model

The source files needed for the FPGA implementation are the following:

```
argo/async_FPGA/src/self_timed_noc_2x2.vhd
argo/async_FPGA/src/router.vhd
argo/async_FPGA/src/channel_latch.vhd
argo/async_FPGA/src/AS_Delay.vhd
argo/async_FPGA/src/AS_bd_2p_latch_ud.vhd
argo/async_FPGA/src/hpu.vhd
argo/async_FPGA/src/crossbar.vhd
argo/async_FPGA/src/AS_C_Generic.vhd
argo/async_FPGA/src/AS_C2.vhd
argo/async_FPGA/src/AS_FPGA.vhd
argo/async_FPGA/src/noc_defs.vhd
argo/async_FPGA/src/delays.vhd
argo/async_noc/src/latch_controller.vhd
argo/async_noc/src/hpu_comb.vhd
argo/async_noc/src/crossbar_stage.vhd
argo/async_noc/src/fifo.vhd
argo/common/math_util.vhd
argo/common/config.vhd
argo/common/ocp.vhd
argo/common/ocp_config.vhd
argo/common/noc_interface.vhd
```

To run simulation both before synthesis and after place&route the Xilinx library models need to be compiled for ModelSim. This is done using Xilinx CompXlib tool. The library models required for pre-synthesis simulation are unisim and secureip. The required library model for the post-place&route simulation is simprim.

To run the simulations, pre-synthesis and post-place&route, scripts are provided under /async_noc_FPGA/sim. The paths in the scripts need to be adjusted in order to access the compiled library models. The scripts can be run in ModelSim environment, as:

```
do pre_synthesis_sim.do
do post_place_route_sim.do
```

6 Requirements

The requirements from Deliverable D1.1 related to work package 3, are satisfied in the same way as reported in Deliverable D3.4 report [3].

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: D3.3 - hardware implementation of the self-timed noc. `http://www.t-crest.org/page/results`.

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

[4] 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.