Rapid prototyping of Networks-on-Chip on multi-FPGA platforms

Junyan TAN[1] , Virginie FRESSE[2], Frédéric ROUSSEAU[3]

[1]*College of IoT, Hohai University, 21300 Changzhou, CHINA*
[2]*Hubert Curien Laboratory UMR CNRS 5516, 42000 Saint-Etienne, France*
[3]*TIMA Laboratory, UJF/CNRS/Grenoble INP, 38000 Grenoble, FRANCE*

Abstract. Experimental approaches used for architecture exploration and validation are often based on configurable logic device such as FPGA. NoC architectures require multi-FPGA platforms as the resources of a single FPGA are not big enough. Partitionning a NoC on multi-FPGA requires special techniques for allocating communication channels, physical links and suitable resource allocation scheme. We present a scalable emulation platform and its associated design flow based on a multi FPGA approach that allows quick exploration, evaluation and comparison of NoC solutions. The efficiency of our approach is illustrated through the deployment of the Hermes NoC and its exploration on several FPGA platforms.

1 INTRODUCTION

Networks on Chip (NoCs) have emerged as a viable option for designing scalable communication architectures for SoC and MPSoCs for signal and image processing applications [1]. However, the design of NoC means making several architectural choices, just like buffer sizing, flow control policies, topology selection. These choices must be made at the design time keeping in mind that the final NoC must stratify a set of critical constraints which depend on the target application such as: latency, energy consumption, design time. The design space being very wide, automation of the design flow and automatic architecture exploration must be considered to ensure a rapid evaluation and test of each solution. Mathematical and experimental approaches are used to accelerate the architecture exploration. Experimental approaches use simulation or emulation with different abstraction levels. FPGA devices are commonly used for emulation and test. Today, several NoC architectures have successfully been implemented on mono FPGA platforms [1][2][3]. One single FPGA device usually does not offer enough resources any more to support a complete large Network/ System on Chip architecture. Such large systems need to be partitioned over several reconfigurable devices, most of time on multi-FPGA platforms for emulation. Existing tool target the mono FPGA platform and there is no any development tool proposed for the multi-FPGA implementation. In [4], a NoC specific multi-FPGA technique is proposed. The major drawback of this work is that the platform was tested with synthetic traffic generators, and authors have not studied the impact of time accuracy loss on real world applications.

Porting a NoC on multi-FPGA platforms means that designers have to manually adapt the architecture with a manual partitioning and to manually integrate inter-FPGA communication blocks. Such adaptations lead to a significantly increased design time.

The aim of this work is to propose a generic design flow for the emulation of large NoC-based MPSoCs on multi FPGA platforms. This design flow integrates a tool that automatically builds the NoC architecture that is partitioned and ready for the multi FPGA implementation. The tool also includes logic blocks to handle the inter-FPGA communications, and all necessary logic required for emulation.

2 DESIGN FLOW FOR THE GENERATION OF THE MULTI-FPGA EMULATION PLATFORM

A new design flow must be proposed to generate the emulation architecture to multi-FPGA platforms. The design flow takes as inputs 1) the NoC architecture, 2) parameterized adaptation blocks: a description of the multi-FPGA platform: the number of available FPGA chips, the type (resources) of FPGA and the physical links used for inter-board communications, 3) emulation blocks: traffic generator and traffic receptors.

2.1 NoC architecture

Network on Chip (NoC) are communication architectures with high scalability, high performance and energy efficient customized solution. NoC architecture is composed of Network Interface (NI), Switch, Links and

Resources. These basic elements are connected using a topology to constitute the NoC architecture.

Data transmitted in NoC architectures are sent through messages. Several data can be sent with one message and one data can be sent with several messages. One message is a set of packets and a packet is a set of flits (Flow Control Unit). The flit is the basic element transferred by a NoC. Packets are sent with idle times specified with data injection rate. Data injection rate is defined as the ratio of the amount of receiving data on its ability to carry data. A 50% data injection rate indicates that the packets use 50% of the bandwidth.

2.2 Adaptation blocks for inter-FPGA communication

Both parallel links and serial links can be used for the intercommunication between FPGA. The high-speed serial link is used as inter-FPGA communication medium in this paper but any other serial or parallel communication can be inserted with the identical methodology. FPGA giants Xilinx and Altera integrate high-speed serialiser/deserialiser (SerDes)[3][5]. High-speed SerDes are made of two functional blocks: the Parallel in Serial out (PISO) block and the Serial In Parallel Out (SIPO) block.

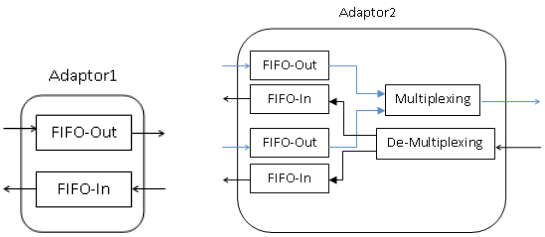

Figure 1. Parameterized IP Block for inter-FPGA communications

Parameterized adaptive blocks are designed for connecting links of the NoC inside the FPGA to IP blocks dedicated to external high-speed serial links. The partitioning depends on the number of physical links (NPL) and the number of inter-FPGA (NIF) links specified. Two scenarios may exist:

1. $N_{PL} >= N_{IF}$: one inter-FPGA link uses one physical link. Adaptations consist in inserting FIFO to adapt frequencies. FIFOs are designed to store one packet.
2. $N_{PL} < N_{IF}$: one inter-FPGA link uses several physical links. In this case, multiplexers, de-multiplexers and FIFOs blocks are inserted between the serial link and the NoC architecture.

Two VHDL parameterized adaptive IP blocks are designed (Figure 1) and inserted in library of the design flow. Adaptor1 contains parameterized FIFOs and is used for scenario 1). Adaptor2 contains parameterized FIFO, multiplexer and de-multiplexer blocks required for scenario 2). These blocks are automatically parameterized according to external and internal frequencies, size of data, number of serial-links and size of NoC.

2.3 Traffic generators

For the emulation of the NoC, IP blocks connected to the NoC are replaced by deterministic traffic generators (TG). These traffic generators simulate the traffic flow between IP blocks inside the NoC with a stochastic traffic distribution to reproduce the behavior of a real IP block. Several traffic generator models have been proposed but none of them are suitable for image and signal processing applications. For example the source address is not given, the number, position and type of data to send either. The proposed TG contains respectively the address of the initiation core, address of the destination cores, the size of transmitted packet, the number of packets to be sent to the destination core, the number of inter-FPGA link crossed.

Several emulations are proposed:

- First emulation explores timing performance according to the data injection rate. The data injection rate is automatically and dynamically generated from a 0% to a 100% load.
- Second emulation is based on a given data injection rate (specified by the designer as a constant value).

2.4 Traffic Receptors

The traffic flow generated by traffic generators is sent through the NoC and then received by traffic receptors. Traffic receptors analyze received packets and extract transmission performances of the NoC. Two types of traffic receptor exist: the statistics from traffic receptor and the trace traffic receptor. Both type of traffic receptors are parameterized VHDL blocks inserted in the design flow.

2.5 Design Flow

The design flow depicted in Figure 2 automatically generates the emulation architecture for mono-FPGA or multi-FPGA platform.

The design flow is based on the existing NoC architecture implemented on a mono-FPGA platform. The designer can insert at the input of the design flow any existing NoC as long as the HDL description is available. From the **classical NoC architecture**, the designer manually selects the number of FPGAs, the number of external serial links and the partitioning of the NoC with a text file containing several fields. The designer associates switches to the target FPGA in the ***partitioning field.*** He indicates how to connect the switches together in the ***connection field***. Then the inter-FPGA communication using aurora blocks are associated to switches in the ***specifying field***. In the example depicted in Figure 3, switches (0,0)(0,1) are implemented on FPGA_0 and other nodes on another FPGA. Switch (0,0) from the first FPGA is connected to switch(0,2) with one aurora communication block. Based on Lex and Yacc, this text file is analyzed, and then, the tools remove the internal connections (links between switches) of the NoC description, inserts parameterized adaptation blocks detailed in 2.2 and serial blocks generated by the

development tool associated to the FPGA from the libraries of the design flow. Adaptation blocks integrate mux-demux blocks and FIFOs that are automatically parameterized according to the size and number of flits to transmit. The partitioning selected implies that the node is associated to its switch. Both cannot be implemented on two devices. It is the reason why the text file contains only information about the switches, not about nodes anymore.

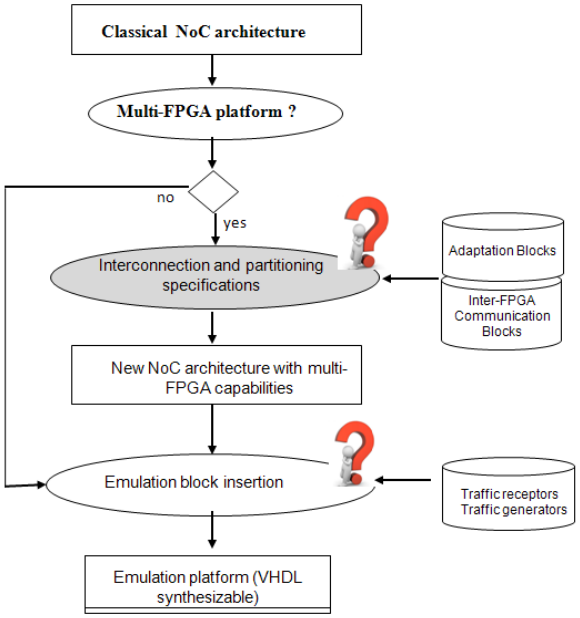

Figure 2. Design flow for the generation of the emulation platform.

The design flow generates a **new NoC architecture with multi-FPGA capabilities** according to the designer requirements. At this step only the communication architecture is generated. Without any text file, the NoC is not partitioned and is implemented onto one FPGA only. In this case, the design flow directly goes to step **Emulation block insertion** presented later.

```
** Partitionning
Switch (0,0) = FPGA_0
Switch(0,1)= FPGA_0
Switch (0,2) = FPGA_1
Switch(0,3)= FPGA_1
**Connections
Switch(0,1)=Switch(0,2)
** Specifying aurora
Aurora_0_in= switch(0,1)
Aurora_0_out= switch(0,2)
```

Figure 3. Example of partitioning specified in the design flow

Then for mono-FPGA or multi-FPGA platforms, traffic generators and traffic receptors are connected to all nodes of the NoC in the **Emulation block insertion step** according to the type of emulation detailed in section 3. This step inserts to all switches TG and TR (by instantiating components declared in the package) whatever the traffic required. The complete emulation architecture on multi-FPGA platform is then generated in HDL description language. This system is synthesized and implemented using existing commercial tools for Synthesis and Place and Route tools inserted in the design flow. The designer implements the complete emulation platform without any hardware requirements.

3 DESIGN FLOW FOR THE HERMES NOC ON XILINX FPGA PLATFORMS

The design flow previously presented is adapted to the Hermes NoC on Xilinx Virtex 5 platforms. Xilinx ISE 14.1 with all integrated tools is used.

3.1 Hermes NoC and ATLAS tool

Hermes is a NoC created by the Catholic University of Rio Grande do Sul (Porto Alegre, Brazil) [1]. This NoC is a 2D packet switched Mesh. The main components of this infrastructure are the Hermes switch and IP cores (Figure 4). The Hermes switch has routing control logic and five bi-directional ports. The local port builds the connection between the switch and its local IP core. All ports possess input buffers for provisional storage of information.

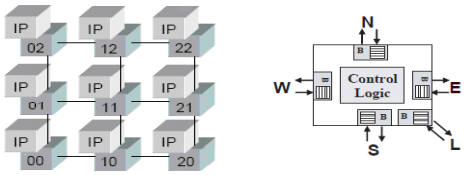

Figure 4. The Hermes NoC and switch architecture

Hermes uses the wormhole flow control. There are several routing algorithms usable in Hermes. The XY algorithm is often used in 2D networks because of its determinism and minimization.

ATLAS is an open source environment designed to automate the generation of the Hermes VHDL structure [6]. Several features can be parameterized in the ATLAS environment: size of flit, buffer depth, number of virtual channels, flow control strategies. These parameters are easily set by the designer to match the specifications of the algorithm.

3.2 Reconfigurable FPGA board for emulation

The multi-FPGA platform used for the NoC emulation is the ML506 evaluation board. Each platform contains a Virtex 5 XC5VSX50 FPGA. The ML506 offers the ability to create high speed serial designs utilizing the RocketIO™ GTP transceivers [7]. RocketIO transceiver with Aurora protocol is used in our emulation platform. It requires a limited design effort to integrate it into an existing design as Xilinx development tools automatically generates the IP blocks dedicated to the serial interface.

Table 1. Utilization of resource on FPGA for different NoC implementations

		Slice Registers	Gain of Slice Registers	Slice LUTs	Gain of Slice LUTs
Mono- FPGA		16% (5260/32640)	X	27% (8831/32640)	X
Multi-FPGA 1 Version1	On one FPGA	10% (3441/32640)	-34%	16% (5356/32640)	-39%
	Total	10% (6882/65280)	30%	16% (10712/65280)	21%
Multi-FPGA 2 Version2	On one FPGA	8% (2898/32640)	-45%	15% (5094/32640)	-42%
	Total	8% (5796/65280)	10%	15% (10098/65280)	14%

3.3 Criteria for emulation on multi-FPGA

The experimental study is mainly based on the average latency, the number of LUTs and registers.

3.4 Emulation platforms of Hermes NoC

The emulations platforms generated by the design flow are based on a 4*3 Mesh. Previous exploration showed that the maximum number of nodes on a single Virtex5 FPGA is 64 nodes. We deliberately use a small NoC to compare results with the mono-FPGA platform. For the multi-FPGA platforms, the design flow splits the NoC, inserts the adaptors blocks and the Xilinx RocketsIO IP blocks. The time for the multi-FPGA emulation platform is several minutes only (depending on the PC and tools used).

Figure 5 and Figure 6 presents the emulation platform on a multi-FPGA architecture with respectively $N_{PL} = N_{IF}=3$ and ($N_{PL\ (=1)} < N_{IF\ (=3)}$). The resources used for all emulation platforms (including the mono-FPGA platform) are depicted in Table 1.

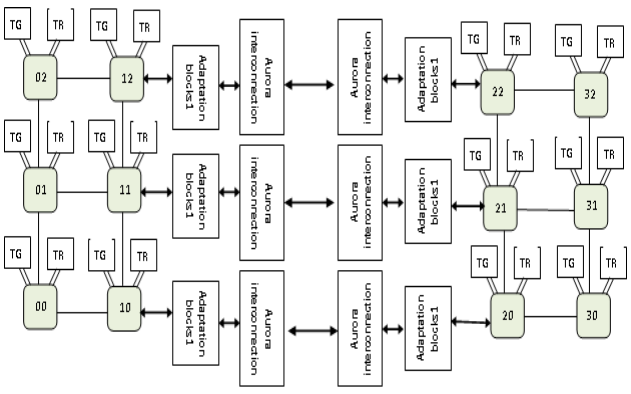

Figure 5. Emulation platform generated by the proposed design flow for the multi-FPGA implementation (scenario 1: $N_{PL} = N_{IF}$)

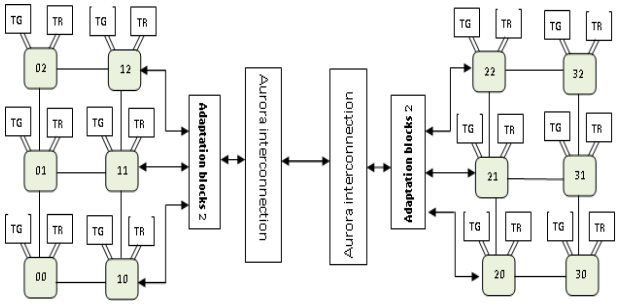

Figure 6. Emulation platform generated by the proposed design flow for the multi-FPGA implementation (case 2: $N_{PL} < N_{IF}$).

The multi-FPGA version1 emulation platform requires 30% more registers and 21% more LUTs than the mono-FPGA emulation platform. The number of registers increases with a factor of 10% and the number of LUTs 14% for version 2 compared to the mono-FPGA emulation platform. With both emulation platforms, the size (number of nodes) of the NoC can be easily extended as the number of resource required for the inter-FPGA communication is low. We assume that the number of nodes depends on the number of FPGA *64 nodes. Moreover, the number of resources for a reconfigurable logic device and for an application specific device can be predicted from multi-FPGA platforms. The most precise evaluation (with 10-15% of resources removed) is from the $N_{PL} = 1$ scenario.

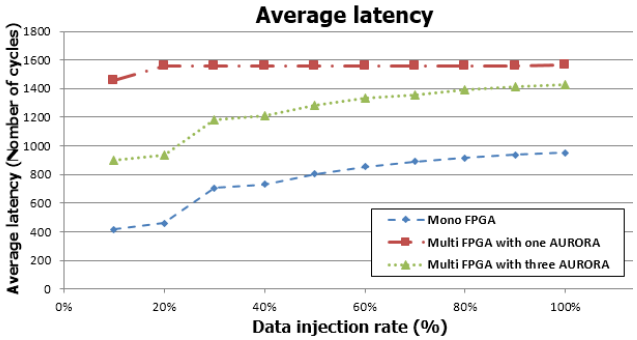

Figure 7. The average latency according to 3 implementation solutions.

The timing evaluation is then based on the sending of 50 packets. Figure 7 presents the average latency for all implementations. For the mono-FPGA scenario, the

traffic receptor receives 15 flits with the average latency of 92 cycles below the saturation point.

The saturation points can be extracted. This point corresponds to the saturation of the architecture. The saturation points are identical for the mono-FPGA and the platform in version1. Multiplexing data for the inter-FPGA communication (version2) changes the position of the saturation point. As the number of inter-FPGA crossed is extracted and specified for each packet, the timing performance can be extracted from case 1. Timing for transmitting packet through the aurora protocol (t_{aurora}) takes 24 cycles for initialization and 1 cycle/flit. FIFOs at the input (t_{FIFO_in}) and output (t_{FIFO_out}) take 2 cycles for controlling and 1 cycle/flit. Therefore it is possible to obtain the timing performance on the mono FPGA by removing these three latencies. For the experiment, the added latency for the inter-FPGA communication is 478 cycles. By removing this time, the end to end latency obtained from case 1is very close to the mono-FPGA implementation by few cycles. The precise timing exploration and the saturation point extraction can be made when $N_{PL} = N_{IF}$. Timing evaluations are more difficult when $N_{PL} <> N_{IF}$.

4. CONCLUSION

A fast and automatic design flow for the implementation of NoC on multi-FPGA platform is proposed. The emulation platform generated is based on a synthesizable code of an existing NoC architecture, high speed serial links and multi-FPGA platforms. This design flow is an "open" design flow: all inputs (NoC, inter-FPGA links or multi-FPGA platform) can be changed or updated. From a partitioning given by the user, the design flow automatically splits the communication architecture, inserts adaptation blocks and emulation blocks and generates the code of the multi-FPGA architecture. Several implementations can be used for resource and timing evaluations.

The design flow generated two types of emulation platform used for timing and resource evaluation. The platform using one link for one inter-FPGA communication will be used for timing estimation (including the saturation point). The platform using one link for all inter-FPGA communication will be used for resource estimation (i.e. to predict the number of resource required for the NoC) and to extend the NoC on multi-FPGA platform in an optimal architecture choice.

ACKNOWLEDGEMENT

This paper is supported by "the Fundamental Research Funds for the Central Universities" (2014B01814).

References

[1] F. Moraes, N. Calazans, A. Mello: "HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip", Integration, the VLSI Journal, vol. 38, no. 1. pp 69–93. Oct. 2004.

[2] C. A. Zeferino, A. A. Susin: "SoCIN: A Parametric and Scalable Network-on-Chip", Proc. 16th Symposium On Integrated Circuits and System Designs, pp 169-174. 2003.

[3] C. Hilton, B. Nelson: "PNoC: a flexible circuit-switched NoC for FPGA-based Systems", in Field Programmable Logic, Aug. 2005. pp 24–26.

[4] Kouadri A., Senouci B., Petrot F., "Scalable Multi-FPGA Platform for Networks-On- Chip Emulation", Application - specific Systems, Architectures and Processors, 2007. ASAP. IEEE International Conf. on , pp.54-60, 9-11 July 2007.

[5] L. Dave: "SerDes Architectures and Applications". Technical Report of National Semiconductor Corporation.

[6] http://www.inf.pucrs.br/~gaph/AtlasHtml/AtlasIndex_us.html

[7] http://www.xilinx.com/