11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN Architectures, Methods and Tools

Large Scale On-Chip Networks : An Accurate Multi-FPGA Emulation Platform

Kouadri-Mostefaoui Abdellah-Medjadji
TIMA Labs
System-Level Synthesis Group
46, Avenue Félix Viallet
Grenoble, FRANCE
Email: kouadri@imag.fr

Benaoumeur Senouci
TIMA Labs
System-Level Synthesis Group
46, Avenue Félix Viallet
Grenoble, FRANCE
Email: senouci.benaoumeur@imag.fr

Frederic Petrot
TIMA Labs
System-Level Synthesis Group
46, Avenue Félix Viallet
Grenoble, FRANCE
Email: Frederic.Petrot@imag.fr

***Abstract*—Interconnect validation is an important early step toward global SoC (System-On-Chip) validation. Fast performances evaluation and design space exploration for NoCs (Networks-On-Chip) are therefore becoming critical issues. A significant speedup of the global validation process for NoC-centric SoCs could be achieved by prototyping such systems on reconfigurable devices (FPGA). However, as SoC complexity increases with the technology scaling, existing general purpose prototyping platforms are far from being suited for large systems. In this paper we present a study for a scalable multi-FPGA platform, designed for NoCs emulation and debugging. This platform allows the integration of complete systems as well as a near cycle-accurate performance estimation.**

I. Introduction

The emerging new generation of SoCs consisting of numerous devices such as μcontrollers, memories, specialized processors, DSPs, mixed signal IPs, increases the communication needs, and such heterogeneous architectures make traditional bus based solutions obsolete. The network-On-Chip paradigm has been now widely adopted by the SoC designers community to overcome communication bottlenecks. Unlike traditional bus-based solution a NoC offers more bandwidth and scalability. However designing with NoCs implies making several architectural choices, such as the switching mechanism, the flow control policies and the buffer planning. Those choices made during design phase must grant that the final NoC satisfies multiple critical constraints such as : minimum latency, energy consumption, reduced design time, etc.

The NoC design challenges make the on-chip-interconnect field an active area of interest, hence several methodologies [5] [7] were proposed in order to automate the whole design flow; those research efforts fall mainly in two categories: formal approaches and experimental approaches. Formal approaches focus on giving a mathematical formulation for the various problems encountered during NoCs synthesis. This kind of problems is generally combinatorial optimization problem and often intractable. The proposed approaches use heuristics, tabu-search, and other approximate algorithms. State-of the art study of NoC synthesis formal approaches demonstrates that this field has reached an acceptable degree of maturity.

Experimental approaches consist of methodologies that are based on simulation and/or emulation tools (SystemC models, reconfigurable devices). These approaches are often used as complementary to the formal ones, in order to validate the theoretical results. However those models suffer from a limitation which is their scalability. In fact simulating a NoC with few cores is feasible in a reasonable time, but for a large-scale NoC including complexes IPs, simulation times become prohibitive.

Emulation of NoC on reconfigurable devices (FPGA) is a very promising way for interconnect validation, as it offers a high speed validation process, and allows for a higher accuracy. However this approach also suffers from scalability issue, mainly due to the resources limitation of reconfigurable devices (equivalent ASIC gates count). Main contributions of this paper are a proposal for a novel multi-FPGA and system-independent emulation platform targeting large scale networks-on-chip; we propose also an approach to precisely estimate the emulation accuracy of the platform for a given system. Our approach has been applied to a complete NoC, and experiments have been conducted to analyze the behavior of the NoC being emulated under various traffic conditions. The rest of the paper is organized as follow: in the next section we briefly discuss some related works. In Section III after the presentation of the generic architecture of the emulation platform we will present in depth its functionalities and the associated synthesis flow. Section IV will give some experimental results and finally, in section V we present the conclusions and discuss future improvements.

II. Related work

Significant research efforts have been done in the Network on chip emulation area, to provide quick performances measurement, and design tradeoffs estimation. Also several HDL simulation environments were proposed in order to avoid the extensive process of physical synthesis.

In [9] the authors describe an FPGA-based platform for NoC infrastructures exploration. The emulated NoC is connected to traffic generators, which are HW components able to generate both stochastic traffic, or regenerate a real-application traffic by reading a prestored traffic profile. The whole system is then synthesized on a FPGA-Chip. This platform makes possible to emulate and explore various NoC topologies, switching policies and other specific architectural aspects. However the proposed platform is not intended for emulating large NoCs and does not allow emulation of the entire SoC i.e. both the interconnection and the cores. In [10] there is also a use of FPGA for a new CDMA-switch technology. [11] presents a NoC architecture based on a parametric and soft-core synthesizable switch architecture, with a wormhole-XY routing. This switch serves a base component for building generic NoC topologies such as 2D-mesh, Torus, or more specific topologies. Others approaches [13] [6], use SystemC rather than hardware emulation, for interconnect validation.

None of the above works provide an effective methodology for complete System/Networks-on-chip prototyping/emulation, as they are focused on a limited set of NoC validation aspects. From our point of view, the most generic approach which may cover all the design aspects passes through multi FPGA emulation. This to allow for a near cycle accurate performances estimation, and a complete system integration.

III. Multi-FPGA emulation of Networks-On-Chip

Typical multi-FPGA prototyping approachs use a technique known as wire-multiplexing [4] for inter-FPGA interconnections, this to overcome the resources and pin limitations. But this technique needs specialized tools; furthermore it slows-down the maximum clock frequency of the emulation. The goal of the present work is to avoid such expensive methods by using high speed serial links to connect multiple FPGA boards. Our approach argues three important advantages:

- Great degree of scalability.
- Most key architectural aspects of a NoC can be assessed (functional validation, topology selection, SW performances, Buffer sizing).
- Very simple configuration and setup.

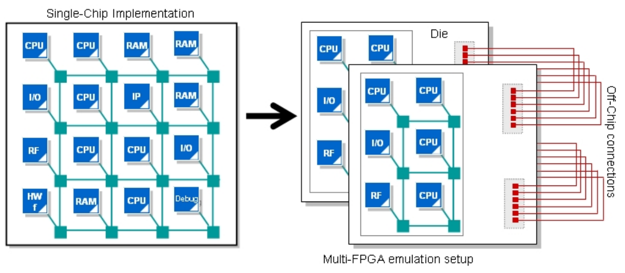


Fig. 1. Multi-FPGA network-on-chip/SoC emulation

Fig.1 shows the global architecture of the platform we propose. The serial links (off-chip links) are intended to simulate physical links between routers (on-chip) of a network-On-Chip. At this point it is worth to mention that the transmission latency of the off-chip lines (about 40 cycles for a 16 bit flit) may degrade the emulation accuracy compared to a single chip implementation of the same NoC. Furthermore, in some cases depending on the NoC topology and board I/O limitations, two

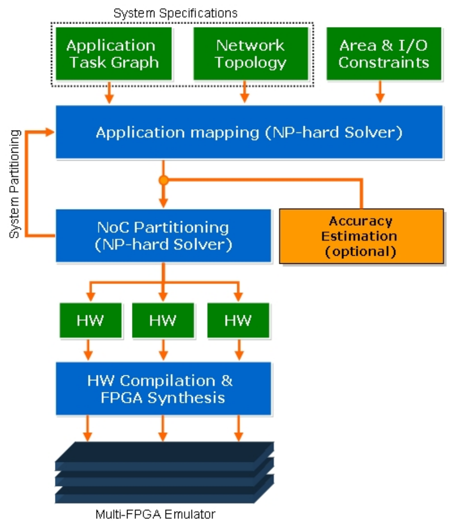

System Specifications
Application Task Graph
Network Topology
Area & I/O Constraints
System Partitioning
Application mapping (NP-hard Solver)
NoC Partitioning (NP-hard Solver)
Accuracy Estimation (optional)
HW
HW
HW
HW Compilation & FPGA Synthesis
Multi-FPGA Emulator

Fig. 2. Multi FPGA NoC emulation synthesis flow

or more physical links can be time-multiplexed on a single off-chip link. This increases the data path latency and the overall “Emulation Error”. The synthesis flow described in the next section is then intended for reducing the emulation error, by choosing the minimum error application mapping and NoC partitioning.

As we stated earlier, the architecture of the emulation platform is extremely scalable. Two essential characteristics contribute to that: The use of serial connections much less bulky than parallel connections. And secondly the number of serial interfaces available on each board (here 8). More generally with n serial interfaces on each board we can build a complete graph having at most (n+1) boards; where each pair of nodes is connected by at least one edge. In other words where the design needs to be partitioned into K FPGA chip, each board must have at least (K-1) serial interfaces. Using the XUP VirtexII Pro board one can build an emulation platform comprising 9 boards; where the maximum distance between any two clusters is 1 off-chip connection.

A. *Multi-FPGA Emulator synthesis flow*

The proposed design methodology (Fig.2) tries to build a customized-accurate emulation platform for a given system, by choosing the right cores mapping and system partitioning; both steps are aimed at minimizing off-chip activity and causing a minimum accuracy alteration.

For speed considerations, the proposed flow relies on SystemC simulation during exploration. Each potential solution (mapping/partitioning) can be quickly evaluated and compared to other solutions without the need of a complete system description, indeed simple SystemC traffic generators can be used in that phase.
We must further point out that the synthesis flow is intended for automation purpose. Hence for some specific cases, it can be outpassed and the partitioning and mapping steps can

be manually performed. An other important point is that in term of functionality, the validation process over the platform strictly respects the topology and the internal mechanisms for data transfer and routing of the original NoC. And then only a subset of performances metrics may be altered during the emulation process.

B. NoC k-partitioning problem formulation

With N reconfigurable devices available, the NoC being emulated needs to be partitioned into K $(K \leq N)$ subsets (cluster) and each one assigned to a reconfigurable device, with respect to the following constraints:

1) Total intra-cluster links must be minimized.
2) A cluster must fit on one FPGA chip.

This problem can be easily reduced to a standard NP-hard graph partitioning problem [12]. The graph partitioning problem is a well-known problem, with numerous applications. In particular in the parallel computing field for load balancing, and in microelectronics for circuit placement, wire routing and chip optimization. Following is the formal definition of a NoC k-partitioning:

- *A NoC is defined by $NoC(V, E)$ a directed and weighted graph where V represents the Network on Chip nodes and E is the set of NoC connections, each weighted w_i (bandwith requirements). In our case NoC is the graph resulting from the mapping step. N represents the total reconfigurable devices available. Let C be the symbolic capacity of each device (Slices, Internal RAM,I/O).*
- *A clustering γ of $NoC(V, E)$ is defined such as :* $\gamma = \{V_0, V_1, V_2...V_{k-1}\}$ *with* $k \leq N$; $V_i \cap V_j = \phi$; $V = \cup V_j$; $0 \leq i, j < k$;
- *The NoC-partitioning problem is then defined by :* ***Minimize***$(\sum w_{ij})$; $i \in V_h$ *and* $j \in V_m; h \neq m$ *with **Ressources***$(V_i) \leq C;$ $0 \leq i < k$
Ressources() returns an estimation of ressources needed by a sub-set V_i, it uses results obtained from a place and route tool (ex: par-ISE for Xilinx devices)

Our flow uses a modified version of a Tabu-search solver from the QAPLIB [2] library for task graph mapping, and the solver used for NoC partitioning is part of the METIS [1] tool suite. Experiments of this solver on large graphs show that it gives high quality partitions compared to other existing algorithms [17]. As an example, -Fig 3- illustrates the partitioning of a 2D-mesh into (3) unbalanced partitions under area and off-chip communication constraints.

At this point we would like to mention that the constraint consisting in minimizing the edges-cut given earlier, leads to build partitions with minimum inter-FPGA connections. However when an estimation of the communication volume for each channel is made available, minimizing the total off-chip traffic rather than the inter-FPGA links minimizes significantly the impact on the emulation precision.

C. Inter-Board Interfaces (RocketIO)

RocketIO [3] (RIO) transceivers are a high speed serial communication interfaces present in the Xilinx FPGA family.

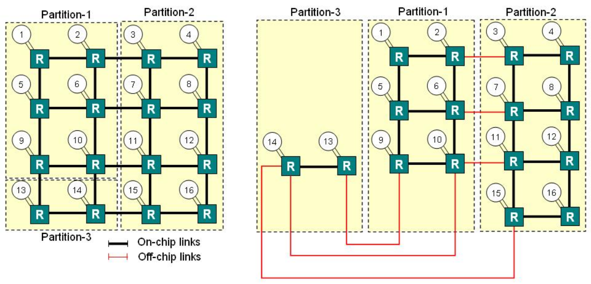


Fig. 3. 2D-Mesh 3-partitioned

A RIO serial interface mainly implements two sub layers:

1) PMA (Physical Media Attachment): Contains the hardware circuitry for clock generation/correction and logic for serialization and deserialization, it also includes all needed buffers for both send/receive operations.
2) PCA (Physical Coding Sub Layer): This layer takes responsibility for data coding and decoding, it includes an 8B/10B encoder/decoder, and also handles the task of cyclic redundancy check (CRC) for transmitted data.

RocketIO Configuration	**Chanels**	**I/O Bit Rate (Gb/s)**
Fibre Channel	1	1.06-2.12
Gbit Ethernet	1	1.25
PCI Express	1	2.5
XAUI (10-Gbit Ethernet)	4	3.125
XAUI (10-Gbit Fibre Channel)	4	3.1875
Infiniband	1, 4, 12	2.5
Aurora (Xilinx protocol)	1, 2, 3, 4, ...	0.6-3.125
Custom Mode	1, 2, 3, 4, ...	0.6-3.125

TABLE I
ROCKETIO STANDARDS PROTOCOLS

Depending on the RocketIO configuration and the clocking scheme being used, RocketIO bit rate varies between 600Mb/s and 3.123 Gb/s [3] (Table I). Although RocketIO transceiver includes a built-in logic for error detection, at 3.125 Gb/s (maximal rate) the bit-error rate tests (BERT [14]) provided by Xilinx indicate that the serial lines are error-free. Consequently there is no need for a particular end-to-end protocol for CRC errors detection and correction.

IV. EXPERIMENTS

In this section we investigate the accuracy of emulation, by first comparing performances of a single-chip NoC implementation with same NoC being emulated on the platform. We used stressing traffic conditions in order to be able to estimates worst case performances. In the second experimentation set we will show how the real performances of a NoC can be derived from the performances collected from the multi-FPGA emulation.

The experimental setup considered the STNoC [8] as NoC benchmark. This interconnect has the nice property to be always 2-partitioned with a fixed external links number (always 4), no matter its size. This property may be exploited in this case to minimize the total number of multiplexed on-chip links, and then minimize the overall emulation alteration (Fig. 5).

The first requirement for building the emulation platform was the need for network protocol adapters. Those components simply adapt the on-chip data-transfer and flow control mechanisms to those of the serial lines considering the timing specificities. Table. II summarizes the FPGA-resources utilization of the protocol adapters, as it can be seen the components require vary small amount of logic. On the other the modular design makes their adaptation to others on-chip protocol very easy.

Cell	Usage	% of Total
Slices	890	6%
Slice Flip Flops	465	1%
4 input LUTs	1400	5%

TABLE II
AREA OVERHEAD FOR THE PROTOCOL ADAPTER ON THE XC2VP30 FPGA

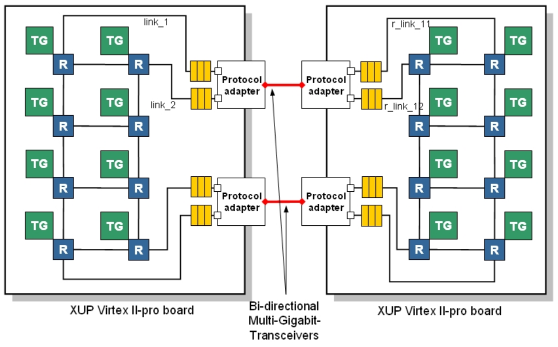
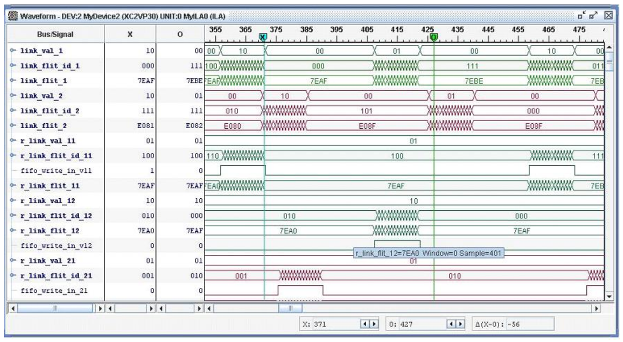


Fig. 4. Experimental Spidergon emulation setup

The alteration notion reflects the accuracy of the platform. It may include several parameters which are complex and inter-related. Consequently this notion can not be defined out of a well-delimited context. It may include : variation of the execution time (for an application), variation of packet latency


Waveform - DEV:2 MyDevice2 (XC2VP30) UNIT:0 MyILA0 (ILA)

Bus/Signal	X	O
link_val_1	10	00
link_flit_id_1	000	111
link_flit_1	7EAF	7EBE
link_val_2	10	01
link_flit_id_2	111	111
link_flit_2	E081	E082
r_link_val_11	01	01
r_link_flit_id_11	100	100
fifo_write_in_vl1	1	0
r_link_flit_11	7EAF	7EAF
r_link_val_12	10	10
r_link_flit_id_12	010	000
r_link_flit_12	7EA0	7EAF
fifo_write_in_vl2	0	0
r_link_val_21	01	01
r_link_flit_id_21	001	010
fifo_write_in_21	0	0

X: 371 O: 427 Δ(X-O): -56

Fig. 5. On-Chip debug waveform trace

through the NoC, variation of the accepted load. The scope of this work being limited to a low level view of NoCs we choose to only address "accepted load variation" and "latency variation". Accepted load gives a general view of the NoC behaviour (congestion threshold), and is defined as follow:

$$Load_{accepted} = \frac{\sum Load_{accepted}(i)}{Total\_Switches\_Count} [\%]$$

$$Load_{accepted}(i) = \frac{Load_{injected}(i)}{Total\_Load(i)} \times 100 [\%]$$

$$Load_{injected}(i) = Switch\ input\ throughput.$$

$$Total\_Load(i) = Traffic\ generator\ throughput.$$

Fig. 5 shows the results obtained considering uniform stochastic traffic sources associated to each node, and by adopting an "*All to All*" uniform address generation scheme.

Initially, different traffic loads were applied, and we measured the accepted load for different buffer sizes (16, 32, 64) Fig. 6 summarizes the results for the initial STNoC model.

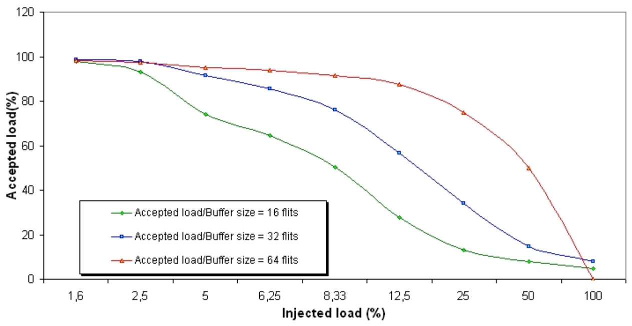


Fig. 6. Spidergon NoC load characterization

The X axis represents the load applied to the NoC (% of maximum load), and Y axis the ratio of effective accepted load. As expected the three curves are ordered according to corresponding buffer sizes, this results serves as base for computing the alteration from the second emulation results.

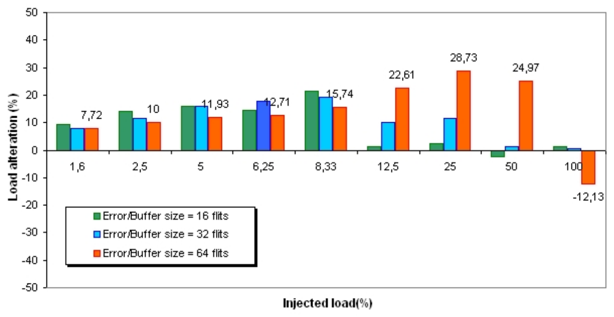


Fig. 7. Estimation of the emulation alteration

Second step is to perform an emulation of partitioned model, under same conditions as first case study i.e.: using the same clock period and the same traffic generators. We obtained results for different buffer sizes of routers, and then difference on accepted load between the two emulations was computed (Fig. 7). Emulation have been conducted with a buffer size for the serial interface of *1024*flit_size*. This has two advantages:

1) It decreases the flow control signalisation for off-chip serial links (sent every *buffer_size* symbols).
2) It increases the accepted load for the overall NoC and then tends to compensate the delays introduced by serial communication latency.

The results shown in Fig. 7 indicate that the error also takes negative values, this does not mean that partitioned NoC has better performance than the original, but results from serial interfaces buffer over-sizing. The maximum alteration for the case-study is about **30%** (worst case). Results demonstrate that the proposed platform does not dramatically affect the emulation accuracy, considering that experimentation scenario can hardly be reproduced in real systems. Especially the uniform traffic model where the communication schedule maintains a constant off-chip activity about **20%** of the overall network activity. However although we used a bursty traffic pattern which is best suited for characterizing the worst case communication delays, we guess that some real-application traffics may cause an error accumulation (such as Ping-Pong traffic schemes or interdependent traffics).

In Fig. 8, is presented the latency measurements for the two NoC models as function of the applied traffic load. The two curves follow the same trends, except that partitioned NoC has a lower critical workload C_1 (from where the latency increases exponentially) than the single-chip NoC. As it can be observed, when the load is less than C_1 in region -A-, the real latency can be directly derived by simply deducing L_d [1] to the measured latency since the behaviours of both NoCs remain relatively linear one from each other. In -B- region contrary to -A- there is no direct method to infer the real latency.

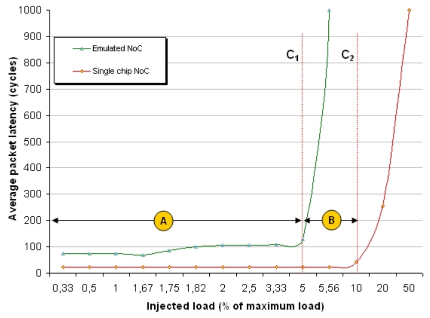


Fig. 8. Emulation accuracy : latency characterization

[1]Generally L_d have to be computed based on the node-to-node latency, and not from the global mean latency of the NoC.

A. The problem of latency correction:

An important question is then to define the conditions under which the emulation process remains in its linear region, where the off-chip latency can be easily bounded. Simply stated, given a traffic model $T(P_s, I_p, B_s, I_b)$ namely $Packet\,size$, $Inter\,packet\,Delay$, $Burst\,size$ and $Inter\,burst\,delay$. The problem we need to solve is to find the correct interval for each parameter such as the latency of the single-chip NoC can be directly inferred. For the sake of simplicity, we consider the abstract model for the serial interfaces depicted in -Fig 9- . Thus the service time for a packet under ideal conditions, where there is no packet contention $Packet_{service}$ can be bounded as follow :

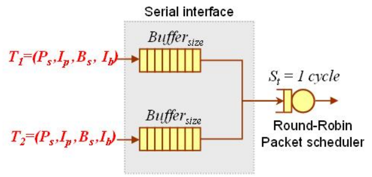


Fig. 9. Queuing model for the two-inputs serial interface

$$P_s \times S_t + 1 \leq Packet_{service} \leq 2 \times (P_s \times S_t + 1) \qquad (1)$$

In (1), P_s is the basic packet size, and S_t is the service time (cycle) needed by the serial interface to treat a single flit. The lower bound of the interval defined in (1) corresponds to the case when the output packet scheduler is in the idle state, then incoming packet can be treated immediately with a delay of 1 cycle (needed for buffering the first flit). The upper bound to the case when an other packet from the other input FIFO is being treated.
Referring to the previous equation, the real node-to-node latency L_r which may be observed in a single chip implementation of the NoC, could be bounded knowing the measured node-to-node latency L_m (Equation.2). Again we should say that this is for the ideal traffic conditions, where no contention on the input FIFOs could occurs.

$$\begin{aligned} L_m - 2 \times M_{st} - S_{id} \leq L_r \leq L_m - M_{st} - S_{id} \\ M_{st}(Minimal\,service\,time) = (P_s \times S_t + 1) \\ S_{id}(Serial\,transmission\,delay) = 38\,cycle \end{aligned} \qquad (2)$$

Given Eq (2), and the simple queuing model of the serial interface, we can further express the conditions of linearity by the following inequality

$$Packet\,arrival\,rate\,A_r \leq Minimal\,service\,rate\,M_{Sr} \qquad (3)$$

Such as :

$$A_r = \frac{1}{I_p + P_s} \; (packet/cycle)$$
$$M_{sr} = \frac{1}{2 \times (P_s \times S_t + 1)} \; (packet/cycle)$$
$$S_t = 1\, cycle$$

At this point, the necessary conditions which ensure a linear behaviour for the overall emulation process can be summarized by the following two inequalities:

$$I_p \geq P_s + 2 \quad (4)$$
$$P_s \leq Buffer_{size} \quad (5)$$

The next step toward experimental validation of proposed analytical model, was performed using an 4X4 2D-Mesh synthesizable instance of the Maia-Hermes NoC [15] and the Xilinx Aurora [16] core for controlling the RocketIO. The analysis was made for the particular case when a single off-chip links is shared between two on-chip traffic flows (-Fig 10-).We intentionally used only two traffic sources in order to precisely control the amount of off-chip traffic. We also assume that packet is consumed immediately once they reach their destination nodes; this is also aimed at giving precise performance estimation, by eliminating delays which may be introduced by a busy destination node.

-Fig 11- shows the node-to-node mean latency evolution as a function of the injection rate ($\frac{1}{P_s+I_p}$), for different traffic scenarios :

1) Two sources S_1 and S_2 are addressing two destinations respectively D_1 and D_2 on a single chip implementation of the Hermes NoC. This result is given as base reference to be compared to the other configurations.
2) The same traffic pattern as (1), but on a partitioned NoC model, i.e. including off-chip communicators.
3) The single source S_1 is issuing traffic to D_1 on partitioned NoC model.
4) Inferred latency corresponding to the results obtained in (2) decreased by the serial lines delays (38 cycles)

The first important point to note, is that the partitioned NoC model leaved his linearity region at exactly $C1 \approx 0.03125\, packet/cycle$, which gives $I_p = 16\, flit$ since $P_s = 16$. This observation confirms the results obtained from the analytical study. A second observation, is that the inferred latency is actually comprised on the analytical bounds given in the previous paragraph ([$L_m - 72$,$L_m - 55$]).

Finally, we would like to point out that although we considered stochastic traffic models for this set of experiments, we guess that the performance deviation will also be very limited for real application traffic, where the impact of the instantaneous traffic events could be much more important on the overall performance estimation.

V. Conclusions & future work

An efficient and scalable platform well suited for large scale NoC-centric systems has been presented in this paper. Our emulation platform is capable of high speed emulation speeds (about 100 Mhz) with a limited impact on the emulation accuracy. The associated design flow allows for automatic partitioning and mapping of systems that do not fit into a single FPGA.

Experimentations were conducted considering two NoCs instances, it confirmed efficiency of the proposed approach. Moreover it pointed out interesting problems concerning accuracy quantification and measurement. This direction will be investigated in our future works aimed at defining new metrics for On-Chip-Networks comparison. To the best of our knowledge this is the first work which pointed-out this concept and gives an effective approach to quantify it.

Finally, as a major issue of the proposed platform is the emulation alteration induced by off-chip communication delays; we will focus on finding the optimal clock ratio (serialization/emulation) that hides completely the off-chip latency and then allow for a near 100% accuracy.

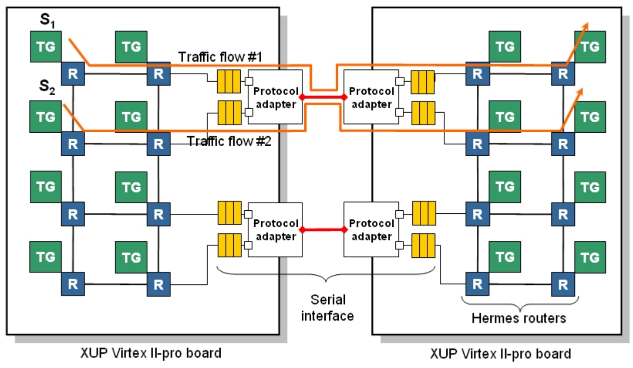


Fig. 10. Hermes NoC emulation using the multi-FPGA platform

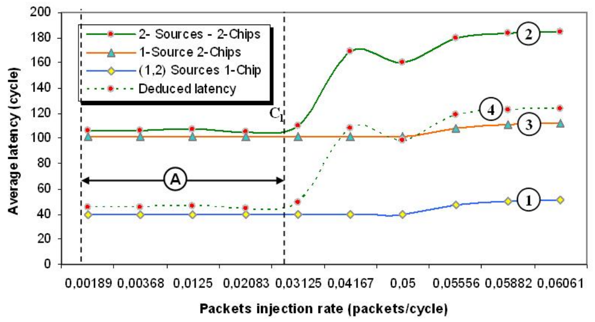


Fig. 11. Latency evolution for Hermes NoC

References

[1] "METIS : Family of Multilevel Partitioning Algorithms". *http://glaros.dtc.umn.edu/gkhome/views/metis*.
[2] "Quadratic Assignment Problem Library". *http://www.opt.math.tu-graz.ac.at/qaplib/*.
[3] "RocketIO Transceiver User's Guide". *http://www.xilinx.com/bvdocs/userguides/ug024.pdf*, pages 21–28.
[4] J. Babb, R. Tessier, and M. Dahl. "Logic Emulation with Virtual Wires". *In IEEE Trans. Computer-Aided Design*, 16(6):609–626, 1997.

[5] L. Benini. "Application Specific NoC Design". *In Proceedings of the conference on Design, Automation and Test in Europe, DATE'06*, pages 1–5, 2006.

[6] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli. "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip". *on the IEEE Transactions on Parallel and Distributed Systems*, 16(2):113–129, 2005.

[7] J. Chan and S. Parameswaran. "NoCGEN: A Template Based Reuse Methodology for Networks on Chip Architecture". *In Proceedings of the 17th International Conference on VLSI Design VLSID'04*, pages 717–720, 2004.

[8] M. Coppola. "Communication Oriented Design Flow". *In ARTIST Workshop at Design Automation and Test in Europe DATE'06*, 2006.

[9] N. Genko, D. Atienza, and G. D. Micheli. "A Complete Network-On-Chip Emulation Framework". *In Proceedings of the conference on Design, Automation and Test in Europe, DATE'05*, pages 246–251, 2005.

[10] D. Kim, M. K. G. Manho, and E. Sobelman. "FPGA-based CDMA Switch for Networks-on-Chip". *In 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines FCCM'05*, 2005.

[11] A. Kumar, I. Ovadia, J. Huisken, H. Corporaal, and J.Meerbergen. "Reconfigurable Multi-Processor Network-on-Chip on FPGA". *In 12th Conference of the Advanced School for Computing and Imaging ASCI'06*, 2006.

[12] A. Lim and Y. Chee. "Graph Partitioning Using Tabu Search". *In Proceedings of the 24th IEEE International Symposium on Circuits and Systems*, pages 1164–1167, 1991.

[13] X. Ningyi, L. Xianglun, L. Renfei, and Z. Zucheng. "A SystemC-based NoC Simulation Framework supporting Heterogeneous Communicators". *In the 6th International Conference On ASIC ASICON'05*, 2005.

[14] "Test Results: RocketIO MGTs with High-Speed Samtec QTE/QSE Connectors and EQCD-EQDP Cable Assemblies". *http://www.xilinx.com/bvdocs/reports/rocketio_hispeed _solutions.pdf*, 2005.

[15] L. Ost, A. Mello, J. Palma, F. Moraes, N. Calazans "MAIA : A framwork for Networks on Chip Generation and Verification". *In Proceedings of the 2005 conference on Asia South Pacific design automation*, 2005.

[16] "Aurora Link-layer Protocol". *http://www.xilinx.com/products/design_resources /conn_central/grouping/aurora.htm*.

[17] G. Karypi, V. Kumar "Unstructured Graph Partitioning and Sparse Matrix Ordering ". *http://citeseer.ist.psu.edu/karypis95metis.html* 1995.