NoC Emulation Based on Partial Reconfiguration

Yana E. Krasteva, Francisco Criado, Eduardo de la Torre and Teresa Riesgo
Centro de Electronica Industrial
Universidad Politecnica de Madrid
Email: yana.ekrasteva,eduardo.delatorre,teresa.riesgo{@upm.es}

***Abstract*—This paper studies the possibly of using partial reconfiguration for achieving acceleration of the emulation process of Systems on Chip based on Networks on Chip (NoC). The main advantage of using partial reconfiguration is that re-synthesis of the systems is not required and thus the emulation process can be accelerated. The paper focuses on the description of a method for building partial runtime reconfigurable systems and its application for building a NoC emulation framework. The paper also includes a brief description of all the building elements of the emulation framework and a use case that demonstrates the advantages of the use of partial reconfiguration for emulation.**

I. Introduction

Future Systems on Chip will contain hundreds of heterogeneous cores that have to be interconnected. To solve the interconnection problem, Networks on Chip (NoC) has appeared as a paradigm shift where packets are routed instead of wires [1] and [2].

Exploring the design space of such complex systems is a very intensive and resource consuming task. To cope with the design of such huge and complex system, a lot of efforts have been put on abstracting the design level of SoC based on NoCs design. Going higher into the abstraction level permits to reduce the design space exploration time and thus the system design time. For a proper NoC selection, some approaches generate NoC topologies getting the application graph representations or application descriptions as starting point. Some examples are SUNMAP [3], based on the Xpipes [4] NoC generator that creates topologies modeled in System C, or in [5], where systems are specified in XML. Other solutions, more accurate, are based on HDL or mixed (System C and HDL), like MAIA [6], where NoC parameters are defined by a user and NoCGeN [7], where the NoC topology can be selected. Lower level, VHDL or RTL, permit to have accurate results, but are more time consuming. Usually, traffic generators and traffic consumer models that simulate real Core behavior are used in order to reduce the simulation time.

On the other hand, there are FPGA based emulation solutions proposed to drastically reduce the simulation and therefore systems evaluation time. For instance in [8], a HW-SW FPGA based emulation framework is presented and combined with the Xpipes environment. Four orders of magnitude of speedup are reported in this work. Emulation, depending on the FPGA available area, may permit to test NoCs using real applications cores instead of traffic models. Some state of the art FPGAs permit to achieve higher flexibility by including partial reconfiguration capabilities, which allows modifying part of a system mapped in the FPGA while the rest, non reconfigured part, is kept active. To take advantage of this powerful feature, that may be included in almost every FPGA in the near feature taking into account the integration level and the needed reconfiguration time, it is necessary to follow a set of steps that permit to create partially reconfigurable systems. The Xilinx solution for building partial reconfigurable systems is presented in [9], while a different approach, slot based, is the solution described in [10]. Anyway, none of the mentioned solution are focused on providing flexibility and cope with problems related to hard cores reallocation along the FPGA structure arrangement

The method, presented in this paper, permits to achieve higher flexibility and tries to make a better use of the on-chip resources. It describes a way for building Virtual Architecture (VA) for partial runtime reconfigurable systems (pRTRS). A VA is defined as a combination of FPGA on-chip resources distribution and on-chip communication strategy definition that permits to build on top of it flexible pRTRS. The method has been applied in the design and implementation of an emulation framework for SoC based on NoCs design. The framework exploits the underlying flexible VA that supports different types of partial reconfiguration, resulting in design space exploration time reduction, because re-synthesis of the entire system is not required. Also problems related to the extraction of data from the FPGA are attenuated.

The rest of the paper is structured as follows. In section II, the used approach to create Virtual Architectures for partially reconfigurable systems based on commercial FPGAs is presented, along with the application of this approach to an NoC emulation framework is described. The entire emulation framework is overview in section III along with the working methodology. Results and a use case are included on section IV and finally conclusions can be found in section V.

II. Virtual Architecture Design

The main goal of the presented Virtual Architecture (VA) design method is to define a set of steps that allow to create reliable and flexible pRTRS, that permit to allocate and securely reallocate a *Hard Core* - partial configuration file, without disturbing the rest - non reconfiguring cores, in different positions in the FPGA. Two main characteristics have been identified and have to be covered by a Virtual Architecture to bring the mentioned flexibility and reliability. The characteristics are listed below:

1) *Core Relocation*. Relocation is important, because permits to define a unique image of the hard core and

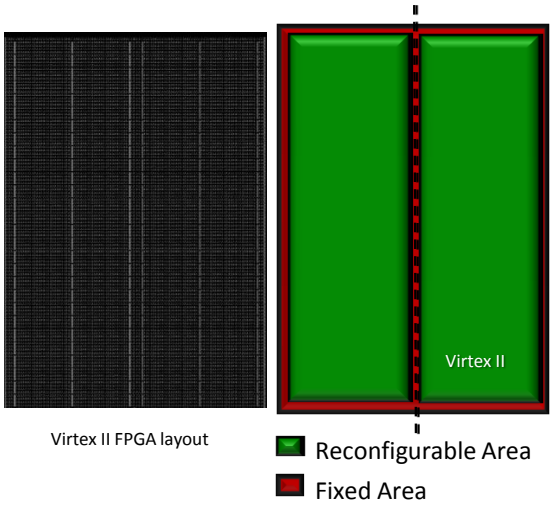

Fig. 1. Virtex II Architecture. General view - left side, Virtual Architecture fixed and reconfigurable areas definition -right side

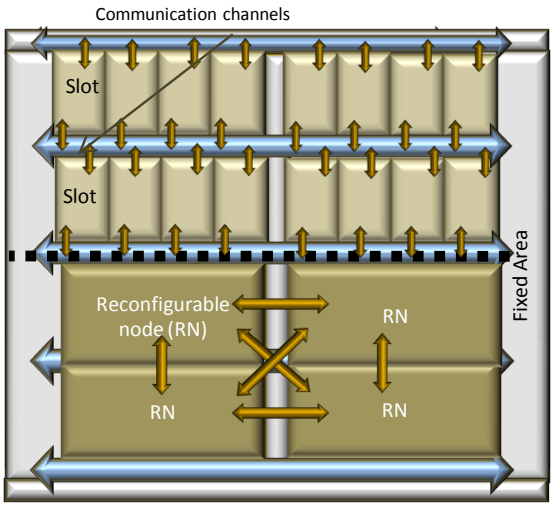

Fig. 2. DRNoC Virtual Architecture. General view in the upper half and the DRNoC mapping on the bottom half.

allocate it at execution time at the proper position in the FPGA. This leads to memory savings and simplifies the reconfiguration control system. Otherwise, as much images of the hard core as possible slot positions have to be kept in memory.

2) The possibility of *loading any size cores*.
3) To have a reliable partial runtime reconfiguration. This means that the on-chip communication infrastructure has to be kept active during reconfiguration. This is not a trivial task because usually this infrastructure is spread accosts the chip.

The design guidelines described next aim to lead to the design of pRTRS VAs that may cover all the characteristic that have been set-up.

1) First of all, the requirements for the pRTRS have to be defined. Regarding VAs, its very important to define the type and the granularity of the reconfiguration or reconfigurations that will be supported. For pRTRS with fine grain reconfiguration (LUTs, a BRAM content and/or FFs), the main part of the reconfigurations are intra-core. For example, to dynamically changed some parameter(s) of a hard core (filter parameters, clock frequency, etc). For this type of reconfiguration, there is no need of defining a VA (macros and/or slot definitions). To successfully perform a reconfiguration in this case, only the exact position of the logic to be change, defined during hard core design, has to be known. This type of reconfiguration is usually called hard core adaptation or small bit manipulation. On the other hand, VAs are needed for medium and/or coarse grain reconfigurations of intra-core or inter-core type and also for core based reconfigurations.
2) Once a partially reconfigurable FPGA that covers the defined area requirements and the reconfiguration method has been selected, the FPGA logic distribution has to be analyzed. The aim at this point is to identify its homogeneity. The specific elements (embedded memories, DSP blocks, multipliers, embedded microprocessors, IOBs, etc.) distribution across the FPGA die.
3) The next step is to define the FPGA internal resource division into fixed and reconfigurable area. The main goal in this step is to select such distribution that the resulting reconfigurable area is as homogeneous as possible. All elements that has a non regular distribution should be assigned to the fixed area. For instance, if the IOBs distribution across the die is regular, then IOBs can be assigned to the reconfigurable area. In the opposite case, they have to be part of the fixed area and to be accessed using special macro structures.
4) Once the fixed/reconfigurable area resource division has been defined, the next step is to divide the reconfigurable area into homogeneous reconfigurable cores, called *Slots*. Slots allocate hard cores. Each slots has to guarantee a hard core access to the maximum amount of FPGA specific resources as possible, as well as the hard core access to the on-chip communication infrastructure. If it is foreseen that all the hard cores in the final application will require access to the same FPGA embedded resources, like multipliers for instance. Then this resource, if it is possible and homogeneity is preserved, will be considered as part of each slot.
5) The last step, but maybe the most important, is to define the internal communication of the pRTRS. The selection of the communication scheme depends on the required flexibility and scalability of the pRTRS. For simple pRTRS, where there is no need of hard core relocation and the amount of slots is kept low (one or two), simple direct connection are the best options (like in the Xilinx approach). On the other hand for robust and flexible pRTRS, bus (like the Hübner approach) and NoC type

internal communication schemes are more suitable. The internal communication also has to guarantee access to FPGA IOBs.

Anyhow, independently of the communication scheme, if flexibility and reliability are targeted, the underlying physical implementation layer has to be able to cope with that. The communication structure has to be localized in a reserved for this FPGA area. This will permit to kept it active during reconfiguration. Also, the occupied by the communication structure area has to be kept as low as possible and access to the communication infrastructure has to be guaranteed for all possible slot positions. Another, very important, aspect of the physical implementation is to carefully select the communication interconnection building elements and the amount of this elements to be used, because this will define the interconnection delay and thus restricts the data transfer rates.

Next a pRTRS VA flowing the method will be presented. The set requirements for the pRTRS to be design, defined in the first step of the method, are that it has to support two reconfiguration types: intra and inter core coarse grain reconfiguration and the the target FPGA, is a Virtex II.
The following steps of the method are described in the next subsection.

A. FPGA Resource Division

During the second step, the FPGA is seen, like usually, as a sea of logic elements whit interleaved, specific, embedded components. For Virtex II FPGA, this elements are: memory blocks (BRAMs), multipliers (MULs) see "Fig. 1" - left side. After the FPGAs homogeneity has been analyzed, in the third step and shown in the "Fig. 1" - right side, a possible resource division in fixed and reconfigurable area is presented. For achieving homogeneity in the reconfigurable area, the fixed zone of the VA includes:

1) The leftmost and rightmost FPGA sides, and also the middle CLB columns. This area can be used to implement modules in charge of internal communication control, as well as external device communication.
2) As there is no homogeneity on the IOBs distribution across the die, the IOBs on the upper and bottom part of the FPGA are assigned to the fixed area.

In the fourth step, the reconfigurable area is divided into homogeneous modules, called *Slots*. A slot is defined as a group of FPGA logic elements prepared to allocate hard cores. slots in this design method have the characteristics listed bellow:

1) For achieving maximum homogeneity, slots can be based only on CLB columns. BRAM, MUL and DCMs are considered separate elements.
2) All slots in a pRTRS have to be *equal* in terms of shape, composition and size.
3) Slots are defined such that hard cores allocated in them have access to on-chip BRAMS and MULs.
4) Slot *does not* span the entire FPGA height. Slot boundaries are restricted by the communication structure borders. This facilitates hard core relocation.
5) A slot accesses the communication structure only through specially defined *access points*, no other communications is available. For achieving relocation, access points have to be placed in the same position relative to slot border, for all slots.

Depending on the slot distribution along the reconfigurable area, two different models can be followed: one dimensional (1D) and two dimensional (2D). With the presented resource division into fixed and reconfigurable areas and slot definition, using a 2D model approach, a VA slot distribution has been defined for the Virtex II family devices. A general view of the Virtex II 2D VA model can be seen in "Fig. 2" - upper half. The model is a mesh of slots, where each slot has access to a general communication structure.

B. 2D based VA Model - DRNoC

Finally, during the last step, the on chip communication has been entirely build by vertical bidirectional LUT based bus macros (BM-LUTs) that pass four wires from slots to the horizontal communication channel and four wires from the horizontal communication channel to slots. Furthermore, communication channels side BM-LUTs input and outputs have been interconnected and routed with the Xilinx ISE PAR tool using only the area reserved for communication channels allocation. This approach, different from the one presented in [11] permits to reduce the communication infrastructure design time, but increases amount of needed resources (LUTs) and the required routing control.
The selected mapping of the NoC based SoC, called Dynamic Reconfigurable NoC (DRNoC), to this general 2D pRTR VA is as follows. Several vertically neighboring slots have been grouped and assigned as a NoC *Reconfigurable Node* (RN), see "Fig. 2" - bottom half. Each RN is connected to all its neighboring RNs with short, hard wired and fixed positioned *communication channels*. Reconfigurable nodes allocate hard cores, network interfaces, routers and/or switching boxes. A router mapped into a RN can use any amount of communication channels and communication channels wires. This approach is similar to the FPGAs CLB local connection, but with higher granularity and is called Xmesh. A DRNoC node is composed of several components: *Reconfigurable element* (RE) or core, *Reconfigurable Network Interfaces* (RNI) and *Reconfigurable Routing Module* (RRM) that include switch matrices. The ideal mapping of a RN is to assign one slot to each RN element. Thus, the presented DRNoC VA will permit to apply reconfiguration at the core and the communication levels and to map any NoC topology (start, mesh or a custom one).
The selected platform for the emulation system is a Virtex II based proprietary board specially created for partially reconfigurable systems design. For mapping DRNoC over the board FPGA, REs and RNIs have been grouped in one slot named also slots and marked with *S* in "Fig. 3", as its function

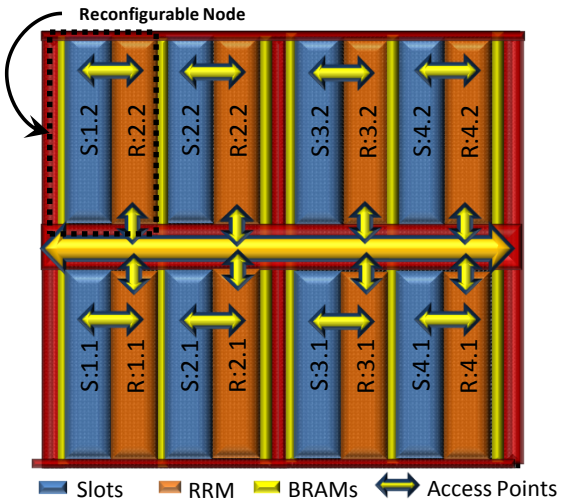

Fig. 3. DRNoC 2D VA General View on an Virtex II FPGA

of mapping IP cores is maintained, while the neighboring slot is assigned for RRMs to be differentiated form conventional slots. In "Fig. 3" RRM are marked simply with *R*. RRMs define the system communication layer, allocating NoC routers, switches, feed-through or buffers. Every core allocated in a slot communicates externally with the RRM only by its access point, thus original vertical access points (see "Fig 2") have been substitute by horizontal access points (see "Fig 3"). Differently, RRMs hold the RNs communication channels control and their connection to the main channel has been maintained. Additionally, each RN row has one measurement bus used for taking data outside the FPGA, all measurement busses are multiplexed in the FPGA fixed area.

The target XC2V3000 FPGA has 56 CLB columns and 64 rows. The selected slot distribution is a 2x4, resulting in a slot size of 144 CLBs, and a 2x2 DRNoC has been mapped on it. The rest of the FPGA is kept for the fixed logic in charge of the communication outside the FPGA. In the middle part of the FPGA, spanning eight CLB rows, the measurement communication structure and the RNs interconnection links are allocated. This VA permits each slot to have access to an embedded FPGA Block RAM column. Also, IP cores, or traffic generator models, as well as routers occupy different areas, therefore RNs can be grouped if it is necessary, to allocate bigger nodes. In this case, the VA hard wire connection links are kept. The pRTRS also permits hard cores reallocation.

The presented DRNoC 2D VA, along with a support software, used to reallocate hard cores vertically or horizontally, permit: i) to allocate a core in any slot position, ii) to group RNs to allocate bigger cores and ii) to dynamically modify the communication strategy and the communication protocol. This designed pRTRs has been used as a based for building and entire emulation platform, presented in the next section.

Finally, DRNoC can be also mapped in latest Virtex 4 and Virtex 5 FPGAs, where block partial reconfiguration is enabled (a block is composed of 256 CLBs).

III. NoC Emulation Platform

The main disadvantages of conventional emulation based solutions are:

1) Synthesis time: every time a system parameter needs to be changed, the system has to be re-synthesized. Another approach is to have all the system options implemented in the FPGA and switch between them, but this consumes a considerable amount of FPGA area.
2) The available FPGA area permits only to emulate relatively small systems. In [12] a solution for this problem is proposed. There, sequentially, parts of a parallel system are loaded into an FPGA. Speedups of 80 to 300 in comparison with System C simulation are reported.
3) System measurement data extraction from the FPGA: the FPGA has much more limited resources in this sense in comparison with a SW based simulation.

By using partial reconfiguration, some of this drawbacks can be attenuated which is the goal of the emulation platform presented in this paper and composed of three parts described next:

1) The NoC design resources along with an associated SW tool for model generation. The design NoC is called Dynamically Reconfigurable NoC (DRNoC) as it has been specially design in a modular way to be mapped over the presented 2D VA for NoCs in the previous section which is the distinguishing characteristic. A set of models in VHDL at behavioral or RTL level used for building and testing different DRNoCs has been created. There are two types of routers available: i) XY routers, based on the HERMES NoC with some minor modification (Buffers are mapped into FPGA BRAMs) and ii) routers based on routing tables (deterministic and adaptive routing with two tables). Regarding the Network interfaces (NI), there are two types - XY and Table. There are also a set of traffic generators (TGs) and traffic receivers (TRs) that are used to model applications behavior. Traffic receivers also include measurement capabilities. It is important to remark that there is a type of traffic receiver that permit to have an online track (in execution time) of the measured parameter. A tool called DRNoCGEN has been created that generates automatically models based on the available resources using a set of user defined parameters.
2) The measurement system is distributed in two platforms: measurement points, included in the DRNoC FPGA (XC2V3000) and the measurement system buffers, based on an XUP board with an XC2VP30 FPGA. Following [13], the system measurement points (a group of measurement registers) are allocated in TRs and in TRs NIs. Data is extracted from measurement points using the AMBA APB interface. The pulled data is buffered in a FIFO allocated in the FPGA area of the XC2VP30 and connected as a custom peripheral to the on-chip Power PC (PPC). The PPC is used to transmit data from the buffer FIFO and send it to a Host PC, through a serial

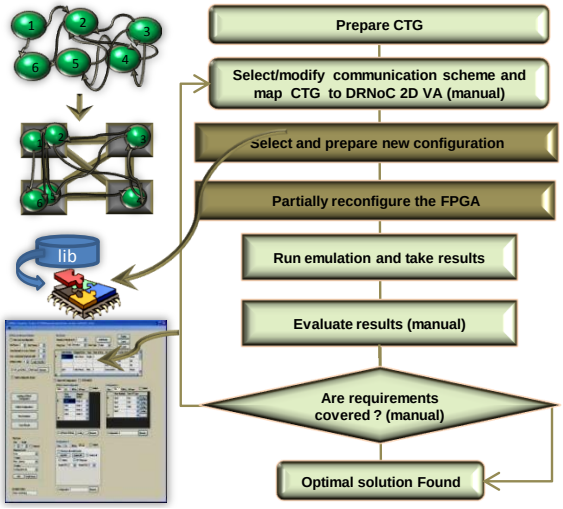

Fig. 4. Emulation Workflow Diagram

port, and also, to control the DRNOC emulation process (run, stop, reset and reconfigure).

3) A SW tool for controlling the emulation process that also holds a hard core library. The SW tool running on the host PC is in charge of controlling the entire emulation process and the design space exploration. The SW includes a GUI and its main features are: i) to define DRNoC configurations and measurement points, ii) to control all the past and future system configurations, reconfigure the FPGA and communicate with the DRNoC measurement system and finally iii) to collect, organize and plot measured data.

The configurations library is composed of partial configuration files that are loaded into the FPGA when required. Partial reconfiguration is used whenever is possible and the required hard core is available in the library. A hard core can be a NoC router, a NI, a TG or a TR. Additionally, smaller partial configuration files are automatically generated from hard cores for intra-core reconfiguration. All this makes this approach very fast as re-synthesis is not needed. The configuration library can be expanded with other hard cores. Anyway, till now, there is not an automated connection between the presented DRNoCGEN tool and the emulation control application. If a new hard core is going to be added to the library it has to be done manually.

A general view of the methodology for NoCs design space exploration is presented in "Fig. 4". The flow begins with an application prepared to be mapped to a NoC, then NoC elements are mapped to DRNoC TR, TGs, NIs and routers and to 2D VA slots and RRMs configuration. Measurement points, NIs and routing modules parameters are defined in the DRNoC emulation SW using the GUI. As much configuration as needed can be defined. After that, the emulation process starts. For each, DRNoC configuration, FPGA configurations for related hard cores, available in the hard core library are automatically arranged, loaded in the FPGA, a single

TABLE I
TR AREA REQUIREMENTS

TR type	Slice	FF	LUTs
TR-Single-online	286	290	395
TR-Single	285	218	528
TR-2	489	350	781
TR-4	810	613	1293
TR-8	1571	1120	2522

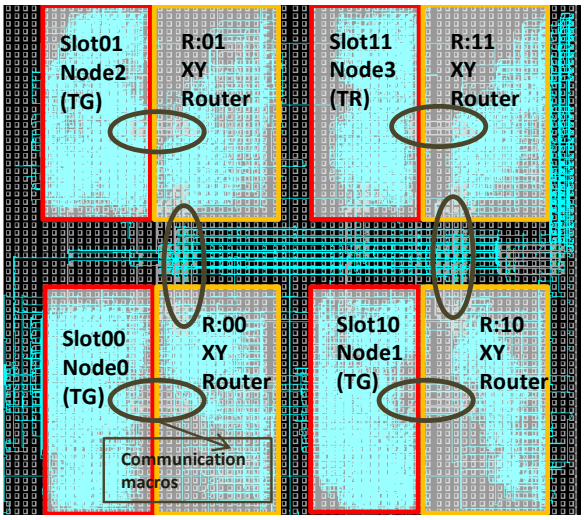

Fig. 5. XY MESH mapped on a 2x2 DRNoC VA FPGA Editor View

emulation runs and results related to each defined measured point are saved. This process is repeated for each DRNoC configuration. Results can be plotted, analyzed and, if needed, new configurations can be added.

IV. Results and Use Case

Area requirements of some TR implementations are presented in Table I, including TRs of both available types: one that follows the online measurement scheme, marked as TR-Single-online, and the remaining TRs follow a max/min value scheme, with different amount of measurement blocks (one measurement block is associated to each TG). For instance, TR-4 tracks data from 4 TGs. It can be noticed that the area needed for a TR increases linearly with the amount of TGs to be tracked. Therefore it is more suitable to have only simple TRs implemented and run the emulation as much times as needed, changing the tracked TG using small bit intra-core reconfiguration.

In the presented 2D VA, the file size for configuring an entire slot requires 181 KB of memory, while for small grain reconfiguration, to change the LUTs configuration, the file needs 4 KB of memory as it includes only the two needed LUT configuration frames. Simply as reference and entire configuration file needs 6MB of memory. the time needed to reconfigure an entire slot is few seconds using the Xilinx configuration tool. Differently, if partial reconfiguration was

not used, the needed time to get with a synthetized systems is in the range of tens of minutes.

For measuring the performance of the entire emulation system, a small example DRNoC implementation on top of a 2D VA has been defined. Since the target FPGA - XC2V3000 is not too large, the biggest VA that can be defined is a 2x2. This is used as a base used for the use case described next. Lets suppose that there is a small application where three sources try to access a common media. Each application source is represented as a traffic generator node (Node0, Node1 and Node2), while the common media is modelled as a traffic receiver node (Node3). This small application has been mapped in two different network topologies. One a 2x2 mesh composed of 4 DRNOC XY routers and a star topology, composed of 3 point to point (P2P) links and one router. The star topology mapping on the FPGA 2D VA is presented in "Fig. 5". Each implementation version (mesh and star) has been emulated with the emulation system. For the mesh topology, emulation time is 1 ms for each desired measurement point, while for the star it is 0,8 ms. For simulating the same system with VHDL simulation, 10 minutes will be needed for the mesh version and 2 for the star. As it has been mentioned, one of the main disadvantages of the emulation is the synthesis time (including place and route). For instance, for the mesh implementation version it takes 16 minutes, while for the star 8 minutes. Differently, for building this system, 2 inter-core partial reconfigurations are needed after the initial, full FPGA configuration and the time needed is in the range of seconds. For instance, if the needed router is not available in the hard core emulation library, only one router is needed to be synthesized and this will take just 1 min. This makes the proposed solution much faster than classical ones.

Results for the online measurement of the traffic received from node0 (TG) is presented for the mesh and star topology in "Fig. 6". Notice that online measurement points permit to stud the network dynamics.

The main drawback of the presented system resides in inherited restrictions of current partial reconfiguration techniques. Although the used method for VAs definition tries to reduce the area overhead due to partial reconfiguration, it is still high. Anyway a tendency of improving the partial reconfiguration capabilities in the newest FPGAs can be noticed. The presented systems can be retargeted to other FPGAs with the exception of the hard core library and the related hard core manipulation SW that support Virtex II and Virtex II Pro FPGAs.

V. Conclusion

As a conclusion it can be said that the benefits of using partial reconfiguration for accelerating emulation process have been demonstrated. The presented method for creating pRTRS based on Virtual Architectures definition brings more flexibility and have permitted to create the emulation platform. Anyway, the restriction of the current technology are high and does not permit to widely exploit the advantaging feature of partial reconfiguration.

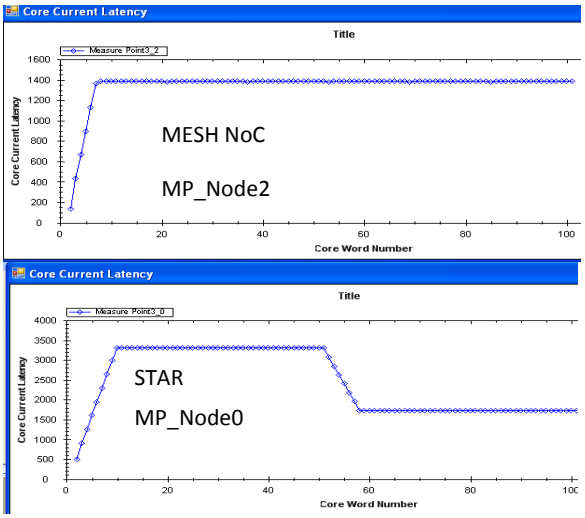

Fig. 6. Plot of the Mesh and the Star Online Latency Measurement Results

Acknowledgment

The authors wish to express their gratitude to the Departamento de Fundamentos da Computacao, Pontificia Universidade Catolica do Rio Grande do Sul, specially to Ney Calazans, for providing the Atlas environment and the HERMES network.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new soc paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] F. Karim, A. Nguyen, S. Dey, and R. Rao, "On-chip communication architecture for oc-768 network processors," in *DAC*, 2001, pp. 678–683.

[3] S. Murali and G. D. Micheli, "Sunmap: a tool for automatic topology selection and generation for nocs," in *DAC '04: Proceedings of the 41st annual conference on Design automation*. New York, NY, USA: ACM, 2004, pp. 914–919.

[4] A. Jalabert, S. Murali, L. Benini, and G. D. Micheli, "xpipesCompiler: A tool for instantiating application specific Networks on Chip," in *Design, Automation and Test in Europe (DATE)*, Paris, France, Feb. 2004.

[5] J. Joven, O. Font-Bach, D. Castells-Rufas, R. Martinez, L. Teres, and J. Carrabina, "xenoc - an experimental network-on-chip environment for parallel distributed computing on noc-based mpsoc architectures." in *PDP*. IEEE Computer Society, 2008, pp. 141–148.

[6] L. Ost, A. Mello, J. Palma, F. G. Moraes, and N. Calazans, "Maia: a framework for networks on chip generation and verification," in *ASP-DAC*, 2005, pp. 49–52.

[7] J. Chan and S. Parameswaran, "Nocgen: A template based reuse methodology for networks on chip architecture," *vlsid*, vol. 00, p. 717, 2004.

[8] N. Genko, D. Atienza, G. D. Micheli, L. Benini, J. M. Mendias, R. Hermida, and F. Catthoor, "A novel approach for network on chip emulation," in *ISCAS (3)*, 2005, pp. 2365–2368.

[9] M. P. Davin Lim, *XAPP 290 (v1.0)0: Two Flows for Partial Reconfigurtion: Module Based or Small Bit Manipulation*. Xilinx, 2004.

[10] H. Walder and M. Platzner, "A runtime environment for reconfigurable hardware operating systems," in *FPL*, 2004, pp. 831–835.

[11] Y. E. Krasteva, A. B. Jimeno, E. de la Torre, and T. Riesgo, "Straight method for reallocation of complex cores by dynamic reconfiguration in virtex ii fpgas," in *IEEE International Workshop on Rapid System Prototyping*, 2005, pp. 77–83.

[12] P. T. Wolkotte, P. K. F. Holzenspies, and G. J. M. Smit, "Fast, accurate and detailed NoC simulations," May 2007.

[13] C. G. et al., "Towards open network-on-chip benchmarks," in *NOCS*, 2007, p. 205.