Virtual Architectures for Partial Runtime Reconfigurable Systems. Application to Network on Chip based SoC Emulation

Yana E. Krasteva, 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*—The paper presents a method for designing Virtual Architectures (VAs) for partial runtime reconfigurable systems (pRTRs). The presented method permits to create flexible pRTRs. Such pRTR system is used as a core for a Network on Chip based SoC emulation. The main advantage of the emulation framework is that it permits fast emulation and design space exploration. The paper includes a brief description of all the building elements of the emulation framework and a use case that demonstrates the advantages of the designed pRTRs.**

I. Introduction

Most reconfigurable systems are linked to FPGAs. FPGAs integration, not only increases constantly, but also extends the spectrum of the embedded elements. Now, state of the art FPGAs include entire platforms in one chip, embedded DSPs and microprocessors, memories and even some include ADCs. Additionally, some of them permit to create highly flexible reconfigurable systems on chip by including partial reconfiguration capabilities, which allows to modify part of the system mapped in the FPGA while the rest, non reconfigured part, is kept active.

In order 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, it is necessary to follow a set of steps that permit to create partially reconfigurable systems. Basically, partial reconfigurable systems can follow two models: one dimensional (1D) and two dimensional (2D). The Xilinx solution for building partial runtime reconfigurable system (pRTRS), based on 1D models, is presented in [1], while a different approach, also 1D, based on sockets and standardized structures (AMBA) to interconnect sockets has been proposed by Kalte et al. in [2]. Further, a similar slot based structure have been proposed by Hübner et al. in [3]. The last two solutions foresee the possibility of grouping sockets/slots to allocate bigger cores. Other solutions are focused on providing relocation along the FPGA structure arrangement. Relocation has been targeted in [4], were relocation, for a 1D pRTRS, directly implemented into the FPGA HW has been presented. Regarding 2D based pRTRS, in [5], Hübner et al. offer an approach to connect/disconnect a core to a vertical communication structure. More recently in [6], Kalte et al., also offer a method for online routing of module interconnections in order to rapidly construct a dynamic reconfigurable system.

This paper tries to generalize the experience grained in partial reconfigurable systems and offers a general method for building and designing flexible partial runtime reconfigurable systems(pRTRS) that permit to reconfigure not only the cores, but also the communication scheme. The method intends to be a scalable solution and targets flexibility and reliability. It copes with the definition and implementation aspects of Virtual Architecture (VA) for partial runtime reconfigurable systems, while making a good use of the on-chip resources. A VA in this paper is defined as a combination of FPGA resources distribution and on-chip interconnection strategy definition. VAs may permit to build flexible pRTRS with different communication schemes. Differently from other approaches, in this one the slots interconnection strategy defined by the VA puts the limits of the design space of on-chip communication schemes that could be defined right before execution and also might be adapted in execution time to current application needs. Regarding the dynamic core insertion in the system, online routing or core connection/disconnection is not foreseen. In our approach, a core could be directly loaded in the systems, if the used communication protocol is compatible, while in the other case, a suitable access method (network interface) has to be added to the core.

The second aspect addressed in this paper is related to Networks on Chip (NoC) emulation. NoCs have appeared as a paradigm shift for solving the interconnection problem in future systems on chip (SoC), where packets are routed instead of wires [7] [8]. Exploring the design space of such NoC based complex systems is a very intensive and resource consuming task. A correct mapping of an application to a heterogeneous SoC by itself is a very complex process. Adding to this the task of selecting an appropriate on-chip communication strategy, makes the system design process even more complex.

To cope with the design of such systems, a lot of efforts have been put on abstracting the NoCs based SoC design level. Going higher into the abstraction level permits to reduce the design space exploration time. In [9], a framework for MPSoC NoC systems modeling, simulation and evaluation, based on System C models is presented. Other solutions are based on HDL or mixed (System C and HDL), like MAIA [10], where NoC parameters are defined by a user and NoCGeN [11], where the NoC topology can be selected. Lower level, VHDL or RTL based models, permit to have accurate results, but are more time consuming. Therefore, in order to reduce the simulation time, traffic generators and traffic consumer models of real cores behavior are used. On the other hand, FPGA based emulation solutions, have been proposed to drastically reduce the simulation and the systems evaluation time. For instance four orders of magnitude of speedup in comparison with simulation are reported in [12]. Another advantage of emulation is that, depending on the FPGA available area, it may permit to test NoCs using real applications cores instead of traffic models. In [13], four real applications mapped into NoCs and prototyped in an FPGA are presented.

The method for VAs design, presented in this paper, has been applied in the design of a pRTRS that has bee used as a core element in an emulation framework for NoC based SoCs design. The framework exploits the flexibility of the VA, that supports different types of partial reconfiguration, to reduce the design space exploration time. Also, problems related to the extraction of data from the FPGA are attenuated.

Partial reconfiguration for FPGA NoC based systems fave been previously used in [14] and in [15]. These papers present two different solutions for dynamically insertion and removal of routers and cores on a mesh based NoC. Also, partial reconfiguration at communication level is used for changing routers buffer sizes in [16].

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. The entire emulation framework is overview in section III along with the working methodology. Results and a use case are included in section IV and finally conclusions can be found in section V.

II. Virtual Architectures 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 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

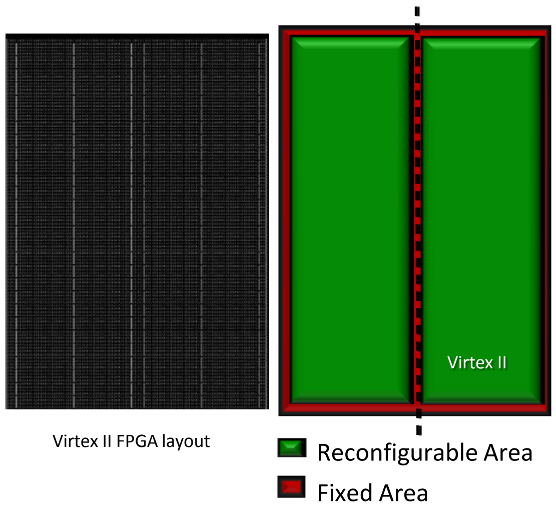

Fig. 1. Virtex II Architecture. General view - left side, Virtual Architecture fixed and reconfigurable areas definition -right side

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 [17] 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

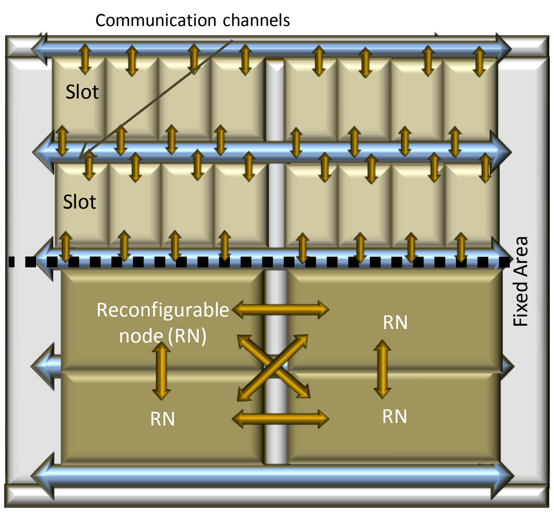

Fig. 2. DRNoC Virtual Architecture. General view in the upper half and the DRNoC mapping on the bottom half.

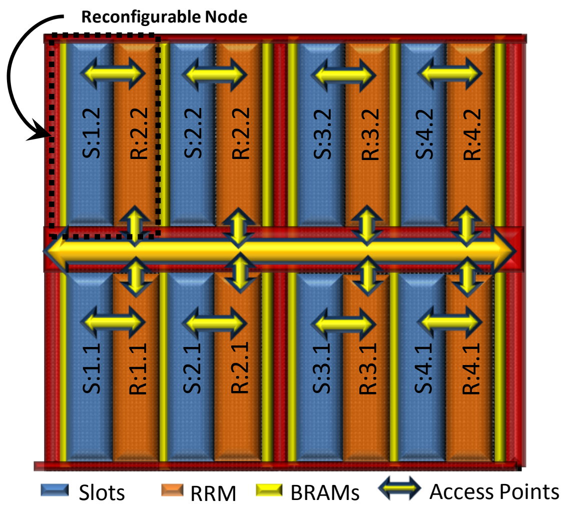

Fig. 3. DRNoC 2D VA General View on an Virtex II FPGA

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 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). But

III. NoC Emulation Platform

The main disadvantages of conventional emulation based solutions are: i) in emulation the system synthesis time (including place and routing) becomes important as the process has to be repeated many times, ii) the available FPGA area permits only to emulate relatively small systems and iii) system measurement data extraction from the FPGA is limited 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

overview 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 designed, modular NoC, is prepared to be mapped over the presented, in the previous section, DRNoC 2D VA which is its distinguishing characteristic. A set of models in VHDL at behavioral or RTL level used for building and testing different NoCs has been created. There are two types of routers available: i) XY routers, based on the HERMES NoC with some modification (buffers are mapped into FPGA BRAMs) and ii) routers based on routing tables (deterministic and adaptive routing with two tables). Regarding Network Interfaces (NIs), there are two types - XY and table based. There are also a set of traffic generators and traffic receivers 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 automatically generates 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 [18], the system measurement points (a group of measurement registers) are allocated in TRs and in TRs NIs. Data is extracted from measurement points using an 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 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, iii) to reconfigure the FPGA and communicate with the DRNoC measurement system and finally vi) to collect, organize and plot measured data.

The configurations library is composed of partial configuration files that are loaded into the FPGA when it is 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 required. 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.

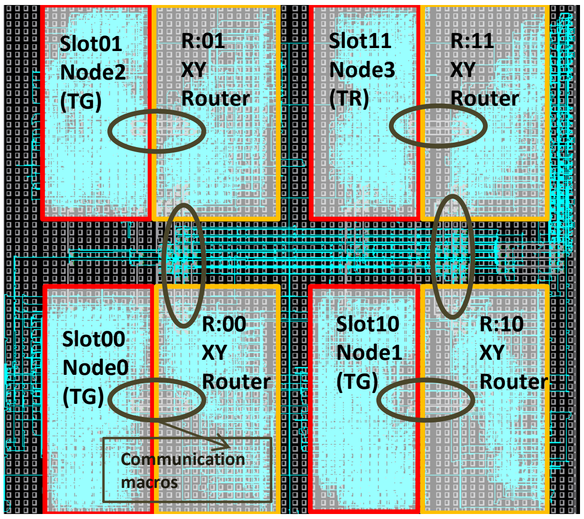

Fig. 4. XY MESH mapped on a 2x2 DRNoC VA FPGA Editor View

The working flow for NoCs design space exploration 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 the 2D VA slots and RRMs configuration. Measurement points, NIs and routing modules parameters are defined in the DRNoC emulation SW using the GUI, and any amount of configuration can be defined. After that, the emulation process starts. For each, DRNoC configuration, FPGA partial bitstreams for related hard cores, available in the hard core library are automatically arranged, loaded in the FPGA, a single 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. The interactive flow continuous until the best NoC configuration has been found.

IV. Results and Use Case

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 1.28MB 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.
On top of, the defined 2x2 DRNoC VA, two small NoC implementations has been mapped and emulated with the

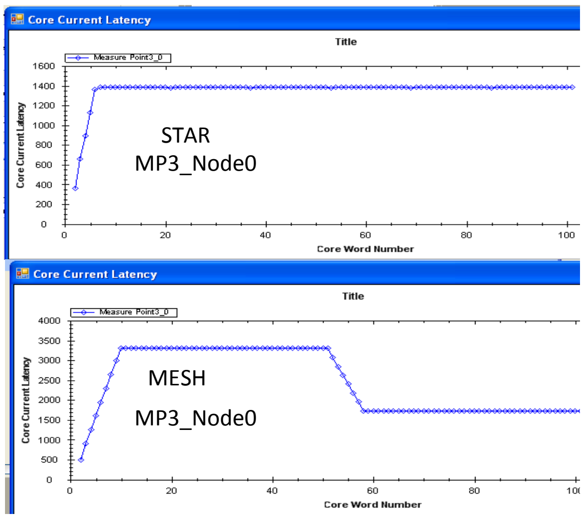

Fig. 5. Plot of the Mesh and the Star Online Latency Measurement Results

emulation system: One is a star topology composed of 3 point to point (P2P) links and one router, while the other is a 2x2 mesh composed of 4 XY routers, while. Four hard cores have been used: three traffic generators mapped from slot 0 to slot 2 and one traffic receiver mapped on slot 3. The mesh topology mapping on the FPGA 2D VA is shown in ”Fig. 4”. Initially, the VA definition file has been loaded in the FPGA. This file contains the internal communication infrastructure, the slots boundaries definition and dummy cores mapped into slots. This forces to define a regular clock distribution along the FPGA. After that, each element of the first configuration has been loaded in the FPGA. First, a suitable hard core is selected form the hard core library, then it is allocated in the defined position and finally, loaded in the FPGA using partial reconfiguration. After that the system is emulated and results are saved in the host PC. For passing to the second system configuration (mesh) to be emulate, only the communication strategy has to be change. This is done by reconfiguring only RRMs. Thus two partial reconfigurations are needed and the sped time is in the range of seconds, differently 16 minutes are needed for synthetizing the second, mesh, configuration.

Results of latency online measurement of a measurement point allocated in node3 that tracks data received from node0 (TG) is presented for the mesh and for the star topology in ”Fig. 5”. Notice that the online measurement permits to evaluate the dynamics of the network.

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

The presented method for VA design has permitted to create a flexible pRTRS that has been used as a core element in the design and implementation of a NoC Emulation Framework. The paper ties to demonstrate the benefits of the use of partial reconfiguration. Anyway, the restriction of the current technology are high and does not permit to widely exploit the advantaging feature of partial reconfiguration.

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] M. P. Davin Lim, *XAPP 290 (v1.0)0: Two Flows for Partial Reconfigurtion: Module Based or Small Bit Manipulation*. Xilinx, 2004.
[2] H. Kalte, D. Langen, E. Vonnahme, A. Brinkmann, and U. Rückert, “Dynamically reconfigurable system-on-programmable-chip,” in *PDP*. IEEE Computer Society, 2002, pp. 235–242.
[3] H. Walder and M. Platzner, “A runtime environment for reconfigurable hardware operating systems,” in *FPL*, 2004, pp. 831–835.
[4] H. Kalte and M. Porrmann, “Replica2pro: task relocation by bitstream manipulation in virtex-ii/pro fpgas,” in *Conf. Computing Frontiers*. ACM, 2006, pp. 403–412.
[5] M. Hübner, C. Schuck, M. Kühnle, and J. Becker, “New 2-dimensional partial dynamic reconfiguration techniques for real-time adaptive microelectronic circuits,” in *ISVLSI*. IEEE Computer Society, 2006, pp. 97–102.
[6] T. F. O. . D. L. Maskell, “Prerouted fpga cores for rapid system construction in a dynamic reconfigurable system,” in *EURASIP Journal on Embedded Systems*, 2007, pp. 548–554.
[7] L. Benini and G. D. Micheli, “Networks on chips: A new soc paradigm,” *IEEE Computer*, vol. 35, no. 1, pp. 70–78, 2002.
[8] F. Karim, A. Nguyen, S. Dey, and R. Rao, “On-chip communication architecture for oc-768 network processors,” in *DAC*, 2001, pp. 678–683.
[9] S. Mahadevan, K. Virk, and J. Madsen, “Arts: A systemc-based framework for modelling multiprocessor systems-on-chip,” *Design Automation of Embedded Systems*, 2006.
[10] 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.
[11] J. Chan and S. Parameswaran, “Nocgen: A template based reuse methodology for networks on chip architecture,” *vlsid*, vol. 00, p. 717, 2004.
[12] 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.
[13] Ü. Y. Ogras, R. Marculescu, H. G. Lee, P. Choudhary, D. Marculescu, M. Kaufman, and P. Nelson, “Challenges and promising results in noc prototyping using fpgas,” *IEEE Micro*, vol. 27, no. 5, pp. 86–95, 2007.
[14] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. P. Fekete, and J. van der Veen, “Dynoc: A dynamic infrastructure for communication in dynamically reconfigurable devices,” in *FPL*, 2005, pp. 153–158.
[15] T. Pionteck, R. Koch, and C. Albrecht, “Applying partial reconfiguration to networks-on-chips.” in *FPL*. IEEE, 2006, pp. 1–6.
[16] R.Benmouhoub and O.Hammami, “Noc monitoring harwdare support for fast noc design space exploration and potential noc partial dynamic reconfiguration,” in *IEEE IES*, 18-20 oct 2006.
[17] 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*. IEEE Computer Society, 2005, pp. 77–83.
[18] C. G. et al., “Towards open network-on-chip benchmarks,” in *NOCS*, 2007, p. 205.