

Hardwired NOC Infrastructure With Integrated Configuration And Functional Architectures

Muhammad Aqeel Wahlah¹ and Kees Goossens^{1,2}

¹ Computer Engineering, Delft University of Technology, Delft, The Netherlands

² Research, NXP Semiconductors, Eindhoven, The Netherlands

aqeel@ce.et.tudelft.nl and kees.goossens@nxp.com

Abstract—In the age of existing platform based MPSOCs where communication among hundreds of Intellectual Properties (IPs) looks eminent, the need for highly scalable and high throughput functional infrastructure is inevitable and no one better than networks on chip (NOC) fulfill that criteria. However in an FPGA the soft NOC constituting reconfigurable switch and CLB matrices takes relatively higher number of logical resources and wiring interconnect thus making FPGAs unfavorable for prototyping intense computational applications.

Through this paper we are advocating a framework for new FPGA on-chip communication architecture which provides a unified mechanism for transporting configuration and functional data. This advocated framework transports configuration data at a rate of 400% higher than the conventional interconnect. Additionally, it provides efficient communication structure for the realization of systems with increased complexities by occupying large number of IP cores.

I. INTRODUCTION

Advancements in process technology have enabled millions of transistors to be placed in such a small area that was never thought before. This immense increase in transistor density has helped Field Programmable Gate Array (FPGA), introduced in 1984 by Xilinx as 1000 gates chip for glue logic purpose, to transpire into million gates chip families of Virtex [1] and Stratex [2] and finding their place in consumer systems not only in the form of glue/ control logic but also in the form of data processing and interconnect logic. Traditionally, an FPGA architecture comprises number of reconfigurable computational blocks and a programmable interconnect. The computational paradigm contains lookup table (LUT) as the basic building block to implement logic functions. On the other hand, FPGA programmable interconnect which occupy most the FPGA area forms the communication paradigm. It entails prefabricated wires of different lengths and switches both used to connect the computational elements to each other. During the recent years, efforts have been put in order to enhance computational capabilities of FPGAs, by introducing embedding microprocessors, DSP units, IP cores and memories. Through these Platform-based FPGAs [1], [2], the vendors like Xilinx and Altera have tried to emulate ASICs performance with the additional advantage of flexibility. These new FPGA chips can be used for not only prototyping but also for realizing complex systems with tens of IP cores with heterogenous characteristics. Additionally, the real time applications pose hard constraints in meeting throughput and latency requirements. Moreover, due to increased complexity of these systems

IP integration along with system verification and validation requires more attention thus increasing the importance of underlying communication architecture which greatly affect the system performance. On the other hand, the programmable interconnect on top of which FPGA communication structures resides although gives unparallel advantage of flexibility but costs in terms of timing closure, high area and high power consumption. In fact, it is this underlying interconnect which is the main culprit for an FPGA system performance to lag considerably behind its counterpart ASIC [3], despite all its embedded blocks and million gates size and will continue to lag unless and until special consideration is given to the FPGA interconnect. Our earlier proposed HWNOC [4] architecture overcomes the limitations of existing functional interconnects through unification of configuration and functional data using the same interconnect. This paper extends the idea of [4] by presenting the details of framework and loading and executing multiple modules of H.264 encoder application through this infrastructure for the approach verification. In order to emulate the real FPGA bitstream loading and execution procedure the framework is built on a cycle accurate transaction level systemC model. It makes use of boot module for IP configuration and a data processor for IP programming and execution purposes. The paper also discusses the impact on number of IP cores with different sizes when compared to the conventional FPGA architecture. The remainder of this paper is structured as follows. The next section presents the related work along with the contributions of this paper. Section III explains the advantages of our proposed HWNOC followed by the step by step realization of our approach in section IV. Finally results are presented in section V from the perspective of isolated and pipelined configuration concurrently with transport of functional data. Results also illustrate benefits in terms of bandwidth throughput and decreased area.

II. RELATED WORK

Different approaches have been used for inter-module communication like point-to-point [5], buses [6], [7], cross bar switches [8], packet and circuit-switched [9], [10], [11], [12] networks-on-chip to enhance gains in area, wire utilization, scalability and re-usability.

Traditionally [5], [6], [7], [8] have been used for data path interconnection because of their simplicity. These interconnects are very good in terms of flexibility and reusability

for lower number of devices but their non scalable, non reusable nature along with insufficient fault tolerance make them susceptible for systems with higher number of processing elements.

Multiple variants of packet-switched and circuit-switched soft NOCs [9], [10], [11], [12], have been proposed to overcome the limitations of above mentioned interconnects in terms of bandwidth scalability and reusability. Since these interconnects are mapped onto the FPGA they pose an extra burden on FPGA power consumption and IP placements due to their relatively higher area and power costs. Only two groups have reported on *hard* NOCs in FPGAs [13], [14], [15] but no hardware architecture details are presented. Moreover [14], [15] propose to use hard NOC as the functional interconnect. Our earlier proposed HWNOC [4] architecture differs from [15] by proposing a unified configuration and functional interconnect and network interface (NI) partitioning in hard and soft regions with clear distinction at the network versus transport layer [4].

III. HWNOC ADVANTAGES

Before entailing the advantages of HWNOC we need to define the terms *hard* and *soft*. A *hard* module i.e. an IP core, BRAM, DSP and multiplier unit is implemented in silicon. We call an IP *soft*, which is independent of the device and after synthesis is mapped on basic FPGA elements like LUTs. HWNOC exhibits number of advantages which make it an inevitable alternative for existing FPGA architecture and few of which are given below.

- 1) Better dynamic reconfiguration. Since proposed architecture does not use the programmable FPGA interconnect therefore it will not impose restrictions on module placement hence enabling them to be placed with more freedom.
- 2) Having this kind of embedded architecture results in placement of higher number of processing elements thus causing greater scalability.
- 3) Some loss of flexibility is obvious due to HWNOC but the cost:performance ratio of 148 times better as compared to using soft NOC [4] indicates that the hard interconnect can be “over-dimensioned” significantly before we lose too much flexibility.
- 4) Since the underlying functional interconnect is not made up of FPGA reconfigurable resources resulting in fewer SRAM switches for placing the same number of IP cores. This causes reduction in parasitic capacitance which means lower power consumption.

IV. UNIFIED ARCHITECTURE

The unified framework initiates by computing the use case configurations at the design time, followed by programming the interconnect to transport the configuration and functional data for candidate IP cores. The details of these procedures are given below.

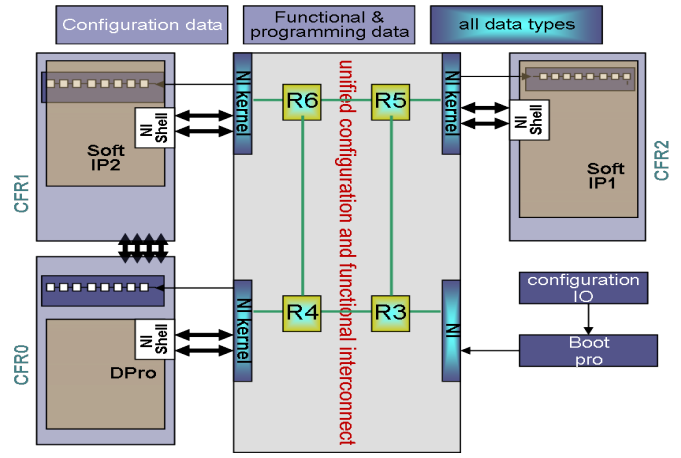


Fig. 1. FPGA with HWNOC.

A. Computation of the configuration

The unified architecture transports both the configuration and functional data for candidate usecase IP cores in isolated and interleaved manner. Since conventional bitstream is delivered to the configuration region as streaming data that must not be interrupted from any other type of data it has stringent latency constraints. The situation becomes even more critical when IP(s) of some real time application are computing data in conjunction with bitstream loading hence demanding guaranteed throughput (GT) in order to stream data between the peers. Our architecture achieves this by providing a GT connection with minimum throughput and maximum fixed latency from the source to destination. In case of a configuration connection boot pro (BPro) and configuration functional region (CFR) serve as source and destination respectively while for functional data it could be any two IP cores. These requirements are calculated statically at design time when a usecase task graph is provided as input along with specified quality of service (QoS) controls i.e. bandwidth and latency requirements for both configuration and data connections [16], [17].

Afterwards the \AA ethereal tool flow, which is chosen as test NOC platform to implement FPGA based NOC, maps the communication cores of a usecase. For guaranteed throughput connections (GT) this results in time slots being allocated by reserving entries in the TDMA slot-table [18].

B. Infrastructure Realization

As shown in figure 1 the proposed hardwired NOC (HWNOC) is connected to a single boot processor (BPro) for bitstream transportation, a data processor (DPro) for sending data to reconfigurable IP blocks, the essential parameters of which e.g. port width and number are decided during the preprocessing phase along with total number of configuration functional regions (CFRs). CFRs divide the FPGA logically into fixed portion of equal sizes. CFRs represent the combination of configuration frames and functional blocks i.e. CLBs, BRAMs and DSP blocks etc. and are connected to

each other as shown in figure 1. Hence making FPGA as a big single functional region however, each CFR is configured through a separate hard NI kernel through which it is attached. Since these regions are not physically disjoint they pose no placement constraints for IP(s) with sizes greater than a single CFR. On the other hand, multiple IP(s) can reside in a single CFR as well. An IP constitutes LUTs and switch boxes which transport data at bit level therefore internal communication within an IP is at bit-level. However, when IPs communicate with each other, they make use of specific protocols (e.g. AXI, DTL) and exchange data in terms of read or write transactions. The proposed HWNOC implements the inter-IP communication at transaction level instead of conventional bit-level. After programming the NOC, the IP configuration, execution of the module is performed through the underlying NOC, the details of which are given below.

1) *NOC Programming*: The proposed architecture needs to be programmed prior to loading bitstream to the corresponding CFRs. In HWNOC each NI kernel has a memory mapped input output (MMIO) port which is used for programming its registers indicating path, time slot allocation and type of traffic i.e. GT or BE. This programming is performed by the controlling agent which in our case is BPro and is achieved by setting up a connection to the remote NI of the target CFR. This involves a request and response channel between BPro and target NI. First the channel to the remote NI MMIO port is set up by writing the necessary registers in BPro. After words same channel is used to set up response channel from the target CFR to BPro. In this manner, the whole NOC can be programmed, i.e. programming and data channels are set up [17].

2) *IP Configuration*: After programming the HWNOC, BPro loads the configuration data of an IP block to the desired configuration functional region (CFR) on the already programmed guaranteed service connection of fixed latency. There is a newly introduced configuration port directly attached to the NI kernel which receives the incoming bitstream frames and with the help of CFR internal logic forwards them to the respective locations. At the end of configuration process the IP is initialized by activating its clock and setting the RESET pin.

3) *IP Execution*: In our design the IP core comprises an embedded NI shell and data path and an IP is ready to receive functional data from the data processor (DPro) after being initialized. The NI shell converts streaming data from the NOC to the IP inputs and outputs. The IP data path then operates on it and outputs data to the NI shell, which forwards it to the next IP or back to DPro on a connection. To avoid overflow between DPro and IP blocks (which may be in different clock domains), the NOC implements credit-based end-to-end flow control [19]. The link-level flow control between the NI shell and the IP used the IP communication protocol's handshaking (e.g. valid/ready for AXI). Hence no data is lost due to buffer overflow, and no invalid data is read due to buffer underflow.

V. RESULTS

We used Xilinx ISE 8.2 to synthesize, place and route the NOC and multiple instances for DCT module of H.264 encoder application onto a Virtex-4XC4VLX200ff1513-11 and used the proposed framework for the same modules bitstream in order to verify the behavior.

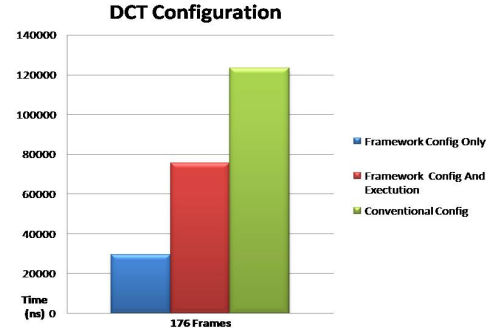


Fig. 2. Configuration Latency.

Results regarding the configuration time for DCT were obtained by transporting DCT bitstream data through our framework, in isolation as well as in conjunction with functional data transportation of other IP cores. Figure 2 shows that the framework requires $0.165\mu s$ in configuration only mode and requires $0.44\mu s$ for pipelined configuration and execution mode (i.e. one DCT instance is executed and other is being configured) as compared to the conventional FPGA which using SelectMAP (32-bit interface at 60MHz) takes $0.7\mu s$ in order for single DCT frame of 41 words to configure the destination region.

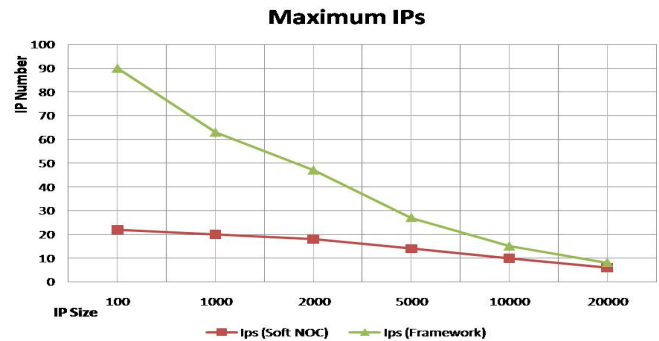


Fig. 3. Total IPs.

The virtex-4 chip contains 178176 LUTs and using the soft NOC (Router-NI occupying 8300 LUT) as underlying functional interconnect could occupy 6 to 21 IP(s) ranging from 20K LUTs to 1K LUTs sizes as shown in figure 3. On the other hand the proposed framework with a soft NI Shell (2K LUTs), varies from 8 to 84 for the same range of IP sizes.

Area comparison of both soft and hard NOC shows that for an IP size of 5K LUTs hard NOC uses 3% of the LUTs as compared to its counterpart soft NOC which uses 42% of the

LUTs. The reconfigurable area occupation by the soft NOC grows for smaller IPs as could be seen in figure 4 where 80% of chip area is used by the soft NOC for 33 IPs of 1K LUT, while the hard NOC costs a reasonable 10%.

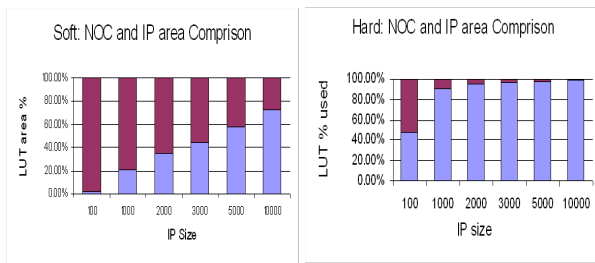


Fig. 4. Area Comparison.

VI. CONCLUSION

Our advocated framework not only replaces the current FPGA configuration interconnect but also uses it for transporting functional data. The proposed architecture reduces timing closure problems in the functional interconnect because it is hardwired and preverified. The advocated architecture achieves not only occupies 3.5 times more IP(s) for a reasonable size of 1K LUTs but also provides configuration throughput of 8Gb/s as compared to conventional 1.9Gb/s. This suggests that the unified framework is better than the existing FPGA configuration interconnect, As well as a good alternative for replacing existing *soft* interconnect.

REFERENCES

- [1] "Virtex-4 Data Sheets," Xilinx, Inc., <http://www.xilinx.com/>.
- [2] "Stratix Data Sheet," Altera, Inc., <http://www.altera.com/>.
- [3] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 203–215, Feb. 2007.
- [4] K. Goossens, M. Bennebroek, J. Y. Hur, and M. A. Wahlah, "Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'08)*, pp. 45–54, Apr. 2008.
- [5] J. Hur, S. Wong, and S. Vassiliadis, "Partially reconfigurable point-to-point interconnects in Virtex-II Pro FPGAs," *IEEE Int'l Conference on applied reconfigurable computing (ARC07)*, pp. 49–60, Mar. 2007.
- [6] C. Bobda, A. Majer, A. Ahmadinia, T. Haller, A. Linarth, and J. Teich, "The Erlangen Slot Machine: Increasing Flexibility in FPGA-based Reconfigurable Platforms," *IEEE Int'l Conference on Field-Programmable Technology (FPT)*, pp. 116–125, Dec. 2005.
- [7] P. Sedcole, B. Blodget, T. Becker, J. Anderson, and P. Lysaght, "Modular dynamic reconfiguration in virtex FPGAs," *IEEE Proceedings Computers and Digital Techniques*, pp. 157–164, May 2006.
- [8] J. Y. Hur, T. Stefanov, S. Wong, and S. Vassiliadis, "Customizing Reconfigurable On-Chip Crossbar Scheduler," *IEEE Int'l Conference on Application-specific Systems, Architectures and Processors (ASAP'07)*, pp. 210–215, Jul. 2007.
- [9] B. Christophe, M. Mateusz, K. Dirk, A. Ali, and T. Jurgen, "A dynamic NOC approach for communication in recongurable devices," *Proc. Int'l Conference on Field Programmable Logic, Reconfigurable Computing, and Applications (FPL)*, 2004.
- [10] Zeferino C.A. and Susin A.A., "Socin: A parametric and scalable network-on-chip," *symposium on Integrated Circuits and Systems Design*, Sep 2003.
- [11] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins, "Interconnection networks enable fine-grain dynamic multitasking on FPGAs," *Proc. Int'l Conference on Field Programmable Logic, Reconfigurable Computing, and Applications (FPL)*, pp. 795–805, Sep. 2002.
- [12] T. Marescaux, J.-Y. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde, and R. Lauwereins, "Networks on chip as hardware components of an OS for reconfigurable systems," *Proc. Int'l Conference on Field Programmable Logic, Reconfigurable Computing, and Applications (FPL)*, pp. 595–605, Sep. 2003, .
- [13] R. Hecht, S. Kubisch, A. Herrholtz, and D. Timmermann, "Dynamic Reconfiguration with hardwired Networks-on-Chip on future FPGAs," *Int'l Conference on Field Programmable Logic and Applications (FPL'05)*, pp. 527–530, Aug. 2005.
- [14] I. Cidon and K. Goossens, "Network and transport layers in networks on chip," *Networks on Chips: Technology and Tools*, ser. The Morgan Kaufmann Series in Systems on Silicon, ch. 5, pp. 147–202, Jul. 2006.
- [15] R. Gindin, I. Cidon, and I. Keidar, "NoC-Based FPGA: Architecture and Routing," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'07)*, pp. 253–264, May. 2007.
- [16] A. Hansson, M. Coenen and K. Goossens, "Undisrupted Quality-Of-Service during Reconfiguration of Multiple Applications in Networks on Chip," *DATE*, pp 954–959, April. 2007.
- [17] A. Hansson and K. Goossens, "Trade-offs in the Configuration of a Network on Chip for Multiple Use-Cases," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'08)*, pp 233–242, May. 2007.
- [18] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to mapping and routing on a network on chip for both best-effort and guaranteed service traffic," *VLSI Design*, May. 2007.
- [19] A. Rădulescu, J. Dielissen, K. Goossens, E. Rijpkema and P. Wielage, "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Programming," *DATE*, Feb. 2004.