Exploring the Performance of Partially Reconfigurable Point-to-Point Interconnects

El Mehdi Abdali*, Maxime Pelcat*†, François Berry*, Jean-Philippe Diguet‡ and Francesca Palumbo§
*Institut Pascal - UMR CNRS 6602, Aubière , France, {el_mehdi.abdali, berry}@uca.fr
†INSA Rennes, IETR - UMR CNRS 6164, Rennes France, mpelcat@insa-rennes.fr
‡Lab-STICC - UMR CNRS 3192, Lorient, France, jean-philippe.diguet@univ-ubs.fr
§PolComIng-UNISS, Sassary, Italy, fpalumbo@uniss.it

***Abstract*—An ever larger share of FPGAs are supporting Dynamic and Partial Reconfiguration (DPR). A reconfigurable point-to-point interconnect (ρ-P2P) is a communication mechanism based on DPR that swaps between different precomputed configurations stored in partial bitstreams. ρ-Point-to-Point (P2P) is intended as a lightweight interconnect that suits the reconfigurable systems where a limited number of configurations are desirable. This paper assesses the pros and cons of ρ-P2P in terms of resource and performance depending on the number of input/output signals, their width and the number of supported configurations.**

Experimental results, conducted on an Intel Cyclone V FPGA, compare ρ-P2P to an equivalently functional non-DPR solution called μ-P2P and to a full crossbar. They show that ρ-P2P is indeed lightweight but introduces performance limitations on operating frequency, memory footprint and reconfiguration time. However, ρ-P2P is in general the least resource intensive of the tested interconnects, except in the trivial case of low numbers of signals and configurations. In particular, an 18×18 full crossbar interconnect requires 75% more resources than an equivalent ρ-P2P. Interestingly, this resource difference between ρ-P2P and a full crossbar grows linearly with the interconnect size.

***Index Terms*—Point-to-Point interconnect, Dynamic and Partial Reconfiguration, FPGA**

I. Introduction

Since the advent of Networks-on-Chip (NoC) in the early 2000s, dynamic networks based on crossbars and buses have been progressively replaced by interconnect fabrics that decompose inter-IP interconnects into different communication layers [1]. However, in applications where the interconnect reconfigurations are chosen among a limited set and operate at a frequency orders of magnitude lower than data processing, a reconfigurable circuit switching interconnect with deterministic latency is an interesting alternative to NoC to connect reconfigurable components in a system [2].

Also introduced in the early 2000s, Dynamic and Partial Reconfiguration (DPR) is a feature available on an ever-wider variety of Field-Programmable Gate Array (FPGA) devices. DPR consists of dynamically changing the logic implemented on a region without affecting contents being executed in other regions of the same device. Many FPGAs currently support DPR, including mid-range devices such as the Cyclone V Systems-on-a-Chip (SoC) from Intel [3]. However, the pace at which industrial applications are adopting DPR is limited [4].

978-1-5386-3344-1/17/$31.00

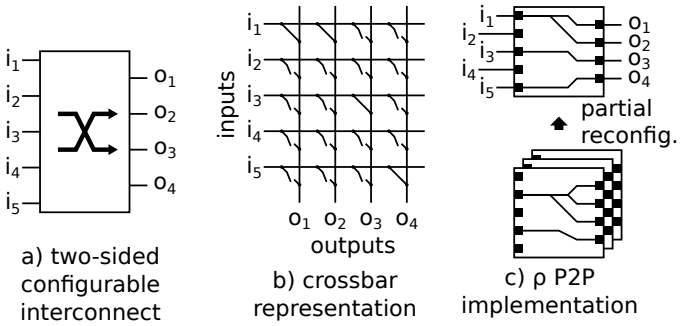


Fig. 1: A 5x4 two-sided interconnect structure and implementations.

The partially reconfigurable P2P interconnect (ρ-P2P) is proposed by Hur, et al. in [2] as a resource efficient reconfigurable network based on DPR. Figure 1 a) illustrates an interconnect with 5 inputs and 4 outputs. Figure 1 b) shows a typical fully deterministic crossbar representation of this interconnect. The full crossbar dynamically connects any input to any set of outputs by closing one switch per column. Figure 1 c) shows a ρ-P2P implementation of the same 5-input $\times$ 4-output interconnect. ρ-P2P is, like the full crossbar, a non-blocking interconnect that associates at runtime any number of outputs to an input without the need for arbitration. ρ-P2P selects one out of K configurations, each of which is stored in a memory in the form of K partial bitstreams, and connects the inputs of the interconnect to its outputs via wires. In comparison with a NoC, ρ-P2P is less versatile but it has been proved to be much less resource intensive [2].

Contributions: This paper reports experimental studies carried out to assess the potential of ρ-P2P to implement a single-stage switch with the objective of building a central interconnect in a DPR-powered system where reconfigurable Processing Elements (PEs) cooperate to perform signal processing in a dataflow way. In [2], ρ-P2P is theoretically introduced and compared to a 2D-mesh NoC with more versatile, but less predictable, communication capabilities. This paper complements [2] by evaluating the performance of ρ-P2P when compared to an equivalently-performing system without DPR

capabilities, named μ-P2P. Our objective is to evaluate the genuine contribution of DPR in the system when the size of the interconnect is varied. Both ρ-P2P and μ-P2P are compared to a crossbar to further assess their performance.

Structure: The paper is organized as follows: related work is presented in Section II. Section III details the experimental setup and Section IV discusses the experimental results.

II. Related Work

Many studies target the dynamic interconnection of DPR-based PEs. As stated in [5], these interconnects are usually either based on buses or on NoC. In [6], a 2D mesh NoC named DyNoC is designed. DyNoC interconnects reconfigurable PEs of different sizes by a large set of distributed routers and wires. Another 2D mesh NoC named CuNoC is presented in [7]. CuNoC uses intelligent routers that support the dynamic placement of PEs by splitting messages into packets and sending packets over several alternative routes. In [5], many bus and NoC topologies are analyzed and a fat-tree NoC named DRAFT is introduced that differentiates some privileged PEs directly connected to the tree root.

The previously presented publications propose to use reconfigurable PEs but the interconnect between them is not based on DPR. The CoNoChi (for Configurable Network on Chip) NoC [8] exploits DPR to add or remove switches from a network at runtime. CoNoChi provides communication for arbitrarily placed hardware modules and potentially reuses processing logic for communication logic and vice versa.

NoC architectures are versatile but ρ-P2P is shown in [2] to be more lightweight than a Packet Switched Network (PSN), at the cost of some flexibility. The reduced resources necessitated by ρ-P2P are explained by its simple circuit switch scheme offering less flexibility than packet switching.

In this paper, the considered interconnects switch between a limited number of configurations, chosen at design time. The motivation for such a hypothesis comes from the current constraints of DPR. Indeed, partial reconfiguration, as of the time of writing this document, is often used to choose, for a few given predetermined FPGA regions, one among a limited set of presynthesized configurations. In a system where P partially reconfigurable PEs are interconnected to solve a given problem, each of which with C configurations and predefined communication schemes per configuration, the maximum number of communication schemes is C^P, and this regardless of the number of inputs and outputs of the interconnect fabric. In this context, the number K of configurations of the interconnect is naturally bounded and a method such as ρ-P2P can be employed.

ρ-P2P offers several advantages over NoC: a fixed and homogeneous latency is guaranteed for each connection, and full Spatial Division Multiplexing (SDM) is offered for a given configuration. Moreover, since the time of a partial reconfiguration is orders of magnitude coarser that the data clock in a system, then a circuit switched network seems promising to manage inter-PE communication in today's DPR

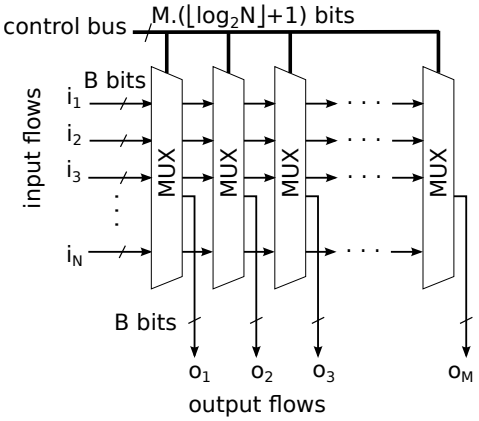


Fig. 2: N inputs, M outputs Multiplexer-based full crossbar interconnect system.

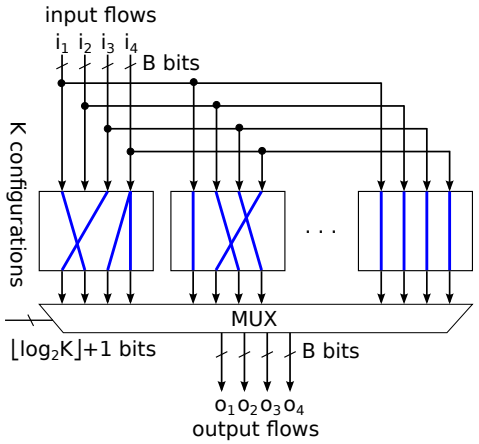


Fig. 3: Connection matrix-based μ-P2P interconnect with $N = 4$ inputs and $M = 4$ outputs.

systems. When the traffic involves long data-stream transmission, circuit switching is preferable to packet switching. In the cases where communication patterns are known at design time, programmable circuit switching can be adopted. As an example, the inter-processor communication protocol implemented by the Pico Array is based on a time division multiplexing scheme, without any need of runtime bus arbitration since data transfers are scheduled on specific time slots managed by software [9]. This approach largely resembles those of programmable crossbars.

Previous publications motivate for the study of a crossbar-like system based on DPR for interconnecting reconfigurable PEs in an FPGA. In [10], Young, et. al generate a full crossbar using DPR with the objective to generate a very large crossbar (928-wire×928-wire). The crossbar is controlled by generating bitstreams at runtime for both Lookup tables (LUTs) and routing. This approach, proposed by Xilinx, is target specific and necessitates to deeply modify the DPR management of the FPGA. In [11], Hoo, et. al consider a reconfigurable interconnect at a LUT level. Authors propose to configure a set of LUTs on an FPGA to form a full crossbar. As in [10],

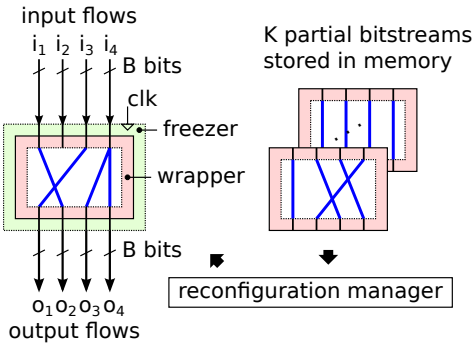


Fig. 4: ρ-P2P implementation with $N = 4$ inputs and $M = 4$ outputs.

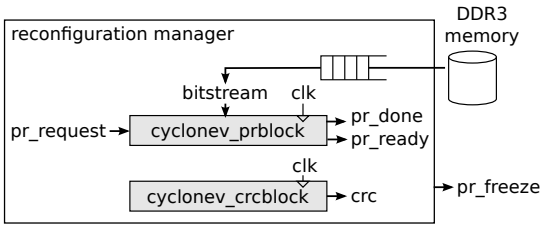


Fig. 5: DPR reconfiguration mechanism via DDR3 external memory

this approach is, by nature, specific to a device and does not translate easily, for instance, to the architecture of Adaptive Logic Modules (ALMs) in Intel FPGAs that combine an 8-input fracturable LUT, two adders, and two registers [3].

Contrary to [10] and [11], our study is conducted in a context where the platform, bitstream synthesizer and DPR machinery are fixed and we intend to study ρ-P2P interconnects within these constraints and at a Register Transfer Level (RTL). As a consequence, the partial bitstreams are generated off-line while in [11] and [10], they must be generated at runtime. To the extent of our knowledge, the performance of ρ-P2P had never been compared to an equivalent non-DPR system. This paper evaluates the benefits of DPR in ρ-P2P, as well as the constraints imposed by the practical use of DPR.

III. Experimental Setup

A. Considered interconnects

This work aims to fairly compare 3 interconnects to assess the real benefits and limitations of DPR-based communication mechanisms. The details of these 3 interconnects follow:

Interconnect 1: A full crossbar of N inputs and M outputs is implemented as a connection grid where each output can be connected to one of the inputs. An output is produced by an N-to-1 multiplexer, each signal being composed of B wires, as depicted in Figure 2. This crossbar structure is likely to be resource intensive and excessive: if the configurations effectively adopted at runtime represent just a small subset of those available, then the instantiated hardware is overestimated. Especially when the real need in terms of particular connection scenarios is small, the full crossbar may become an oversized solution. An alternative that can resolve these issues is to implement only the desired interconnection scenarios. **Interconnect 2:** We introduce the **μ-P2P interconnect**, illustrated in Figure 3, as an alternative to a full crossbar. μ-P2P implements a subset of K interconnect configurations via connection matrices. The selection of the current connection matrix is performed via M K-to-1 Multiplexers (MUXes), where each signal is composed of B wires, M is the number of outputs and K the number of configurations.

The most resource demanding part of μ-P2P is the output multiplexer. It performs the swapping between the connection matrices in order to establish a particular mapping required by the system. Following the same principle, ρ-P2P exploits hardware resource time-multiplexing offered by DPR to swap between the mapping configurations without having to implement them all at the same time.

Interconnect 3: A ρ-P2P interconnect. As depicted in Figure 4, the reconfigurable region of a ρ-P2P interconnect contains only the currently needed connection matrix. The alternative matrices, forming the configuration set, are stored as bitstreams in an external memory. When the system requests a new interconnect matrix, the reconfiguration controller implements the appropriate bitstream file at the cost of reconfiguration time which is the time needed to write the file to the configuration memory and to prepare the reconfigurable region. Contrary to what their names suggest, ρ-P2P and μ-P2P are capable of managing both point-to-point communication and point-to-multipoint.

Measurements of performances for each of the 3 interconnects are conducted under the hypothesis that the communication mechanisms being tested are not supporting bufferisation with First In, First Out data queues (FIFOs). Instead, all output signals from PEs are directly connected to input signals.

B. Dynamic Partial Reconfiguration

In this exploration study, a DPR management system is designed that connects to the Intel Cyclone V DPR hardware. For this purpose, proprietary components from Intel, called *cyclonev_prbloc* and *cyclonev_crcblock*, are instantiated and a manager is created that retrieves partial bitstreams from an external DDR3 memory, bufferizes them and provides them to the *cyclonev_prbloc* component. These components are illustrated in Figure 5 and the signals that drive reconfiguration are shown on the chronogram of Figure 6. Cyclone V DPR is functioning at up to 100MHz on a 16-bit bus, reaching a reconfiguration throughput of 1.6Gb/s. The N 16-bit bitstream words must be provided successively and without discontinuance at the input of the *cyclonev_prbloc*. When implementing ρ-P2P over a Cyclone V FPGA, technological constraints must be taken into consideration. First, a *freezer* component must, during reconfiguration time, force to $'1'$ all signals except clock that input the region. The signal to trigger the freezing process is called *pr_freeze*. Second, as the signals connected to the region must be non-reconfigured, a *wrapper* component must map the external signals to the effectively used ones inside the region, discarding unused signals. The

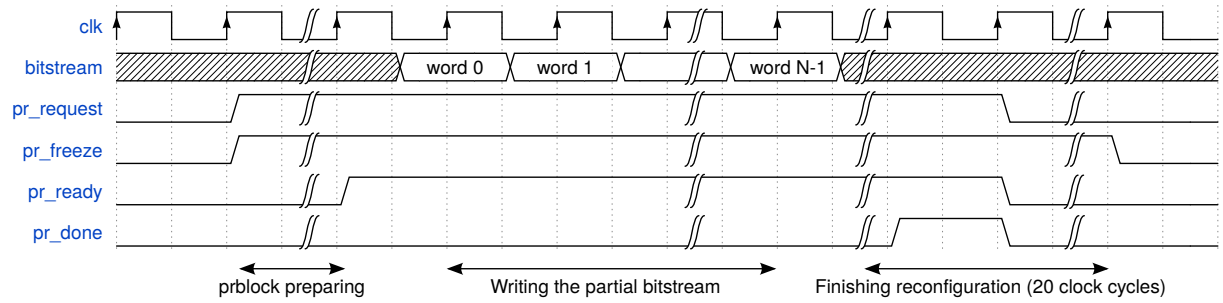


Fig. 6: Chronogram of a partial reconfiguration from a bitstream composed of N 16-bit words.

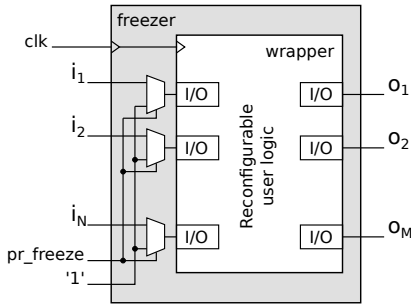


Fig. 7: Interface of a reconfigurable region.

interfacing model of a reconfigurable region is depicted in Figure 7. When a reconfiguration is required, the configuration manager asks for a signal freeze and starts to write bitstream into the configuration memory using *cyclonev_prbloc* while the file integrity is checked using the *cyclonev_crcblock*. When configuration is finished, signal *pr_done* is issued by the *cyclonev_prbloc* and reconfiguration manager must wait 20 clock cycles before unfreezing signals and letting the newly implemented design start to operate.

The Cyclone V FPGA is divided into block columns, each block with one type among three: Logic Array Blocks (LABs), M10K memory blocks and Digital Signal Processing (DSP). To support partial reconfiguration, a region must have a fixed size and position, reserving all the resources inside it. The region is then defined as a partition to activate incremental compilation, and as a *LogicLock region* to force components to fit into the region. Leveraging on time multiplexing, reconfigurable *LogicLock regions* may implement, over time, more logic than non-DPR ones. However, parameters related to region size and resources organization need to be chosen with care [12]. Moreover, the maximum operating frequency of a DPR region is lower than the one of a non-DPR one, as will be discussed in Section IV. Under the Intel toolset, the whole design is duplicated into several *reconfigurable revisions*. Different partial bitstreams, called *personas*, are generated for each region, one per reconfigurable revision. The specific FPGA used for this study is a medium-range FPGA of type Cyclone V 5CGXFC7 embedding 5.6K LABs, each with 10 ALMs for a total of 56,480 ALMs.

The cost in terms of bitstream size and partial reconfiguration duration is roughly proportional to the width of the region and does not depend much on its height. An appropriate strategy is thus to define vertical reconfigurable regions with the lowest possible number of columns. One of the limitations in using DPR is related to internal memory consumption. Indeed, memory blocks, located out of a reconfigurable region R but belonging to columns overlapping with R cannot be used as initialized Random Access Memory (RAM) or Read Only Memory (ROM) [13]. They can only be used as uninitialized RAM.

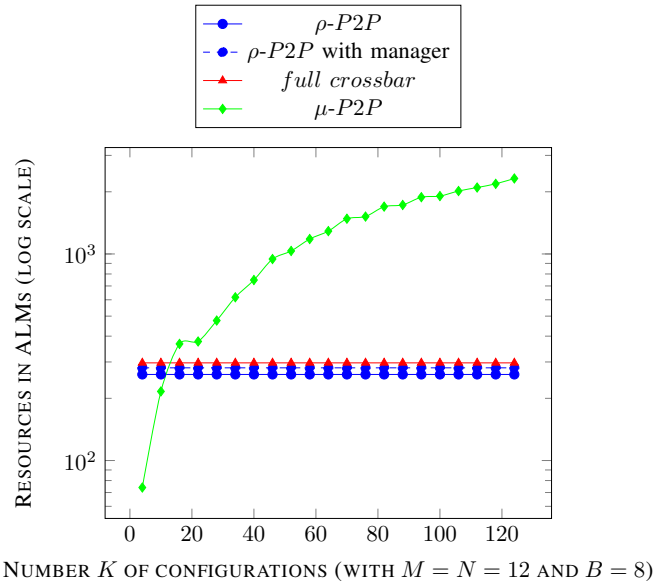
In Section IV, the interconnects are compared in terms of performances by varying $N = M$, K, and B. The metrics used for comparing them are the maximum frequency that translates directly into interconnect throughput, and the resource cost in terms of ALMs. Energy has not been considered in this first study. A fair energetic comparison of ρ-P2P would require taking into account the energy of writing the configuration memory and this concern is kept for future work. In order to estimate as accurately as possible the maximum frequency under which the communication performs, the input clock for the communication systems under test has been separated from the clock driving secondary peripherals and components.

IV. Experimental Results

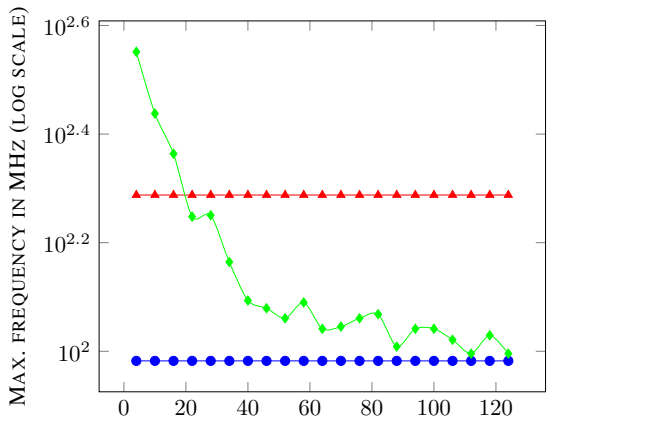
A. Choosing between ρ-P2P, μ-P2P and full crossbar based on resource needs

Figure 8a shows the amount of ALM resources needed to implement an interconnect with 12 inputs and 12 outputs, each composed of 8 bits. The resources for the full crossbar do not depend on the number of configurations K and the full crossbar provides all $N.2^M = 12.2^{12} = 49,152$ configurations. The resources needed by ρ-P2P are also independent from K because configurations are stored in memory. Figure 8a displays ρ-P2P resources with and without the overhead due to the DPR manager. The manager overhead is paid once, even when other elements such as PEs use partial reconfiguration. The resources of μ-P2P strongly depend on the number of configurations.

When K is higher than the input signal number N, μ-P2P requires more hardware resources than the full crossbar. This is explained by the size of the MUX needed to generate the μ-P2P and full crossbar outputs. The full crossbar is implemented by $M \times (N-1)$ 2-to-1 MUXes while a μ-P2P

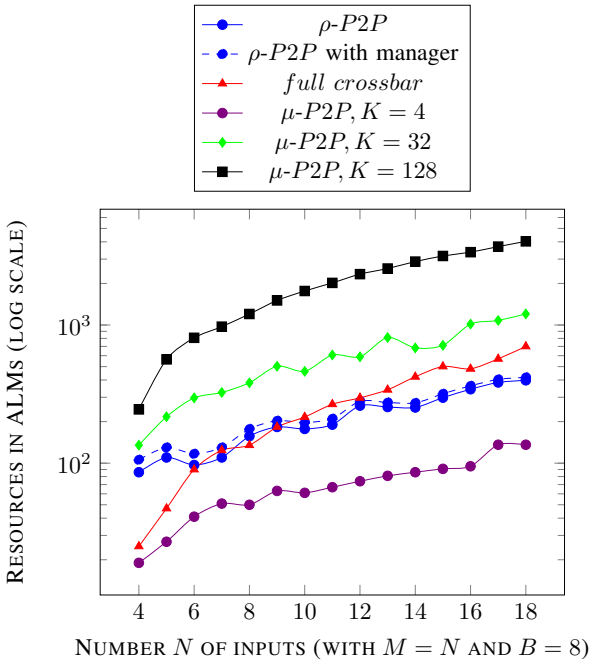


(a)

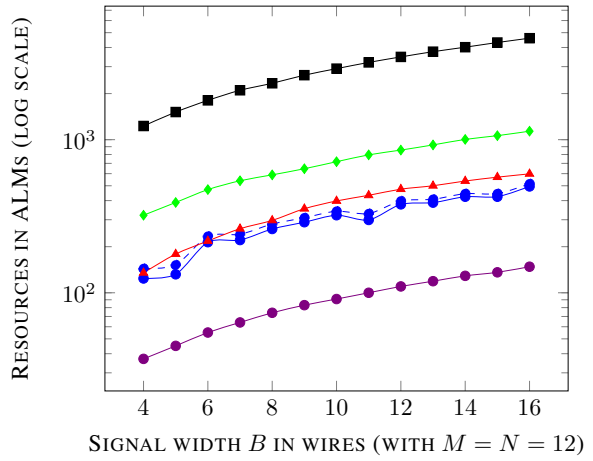


(b)

Fig. 8: (a) Hardware resources and (b) Maximum operating frequency as functions of the number of configurations K.



(a)



(b)

Fig. 9: Hardware resources, (a) as a function of the number of input signals N with $M = N$ (symmetric interconnect case), and (b) as a function of the signal width B.

is implemented using $(K-1) \times M$ 2-to-1 MUXes. Signal width B does not affect previous remarks. Figure 9b displays resources that, as expected, are roughly linearly dependent on signal width B for all interconnects.

Figure 9a shows that when $M = N < 8$, the full crossbar is less resource intensive even than ρ-P2P. As a conclusion, for very small interconnects, there is no benefit in adopting a ρ-P2P interconnect. If $M = N \geq 8$, ρ-P2P is the most resource efficient solution, followed by the crossbar and μ-P2P. The case of very small K is an exception where μ-P2P requires very low amounts of resources.

As a conclusion, the least resource intensive interconnect is ρ-P2P, except when the number of configurations is very low ($K < 8$) and makes μ-P2P the least resource intensive solution, or when the number of input/output is very low ($M = N < 8$) and makes the full crossbar the most optimal solution. Interestingly, the ρ-P2P resource cost increases linearly with a trend of $22.6 \times N - 28$ ALMs while the crossbar resource cost has the quadratic trend $1.8 \times N^2 + 4.9 \times N - 15$ ALMs. Consequently, for $M = N = 12$, ρ-P2P saves only 11% of the resources when compared to a full crossbar (94% if the overhead of the reconfiguration manager is counted) but ρ-P2P becomes more interesting for larger interconnect sizes. For instance, full crossbar requires 75% more resources than ρ-P2P for $M = N = 18$ (69% when considering reconfiguration manager overhead) and the overcost can be extrapolated to 170% for $M = N = 30$.

B. Choosing between ρ-P2P, μ-P2P and full crossbar based on frequency constraints

When considering frequency as the optimization objective, conclusions differ. Figure 8b illustrates the case when $M = N = 12$ and $B = 8$. The synthesizer limits ρ-P2P to $96MHz$ while the full crossbar can operate at up to $194MHz$. The frequency of μ-P2P depends on the number of configurations K. This frequency limitation is due to the combinatorial critical path which is the longest time needed by the gates to compute an output when any of its related inputs change. This path normally relates to the number of crossed gates. The number of stages of the critical path of μ-P2P is $\lfloor \log_2(K) \rfloor + 1$ while for a full crossbar it is $\lfloor \log_2(N) \rfloor + 1$. Thus, when $K < N$, μ-P2P has a larger maximum throughput than a full crossbar. In other cases, the full crossbar is more efficient than μ-P2P and the maximum frequency of μ-P2P and ρ-P2P are equivalent for large crossbars.

C. Discussion on reconfiguration costs

All presented configurations of ρ-P2P fit within one column of LABs on a Cyclone V FPGA. The storage of bitstreams for this column necessitates 160 kBytes per configuration. The needs in terms memory access and the reconfiguration throughput lead to a reconfiguration time of about 1.6ms per reconfiguration. This makes ρ-P2P interesting only if the configuration is not changed frequently. The problem of memory consumption also becomes crucial when the number of configurations increases because a large number of configurations translates into a large amount of memory dedicated to bitstream storage. In a real-life DPR-based system, external memory is required for storing the large bitstreams.

Another limitation of DPR-based systems concerns the overhead in terms of frequency and resources. They are both explained by the necessity for the fitter to hold reconfigurable region signals in fixed hardware locations, and this for all considered personas (configurations). To do so, the Intel Quartus Prime software instantiates automatically a dedicated LUT called wire-LUT for each port of the reconfigurable region to lock down the same location for all persona instances [13]. The added logic has an overhead on the needed hardware resources as well as on the operating frequency since the critical path is extended when new gates are added in the logic chain. This fact explains the limited frequency in Figure 8b at which ρ-P2P can operate when compared to the other communication mechanisms, despite the fact that the design implemented inside the region is composed only of direct signal connections. As a consequence, when DPR is used in a design, the parts that are selected to be in a reconfigurable region will operate at a constrained frequency when compared to the same design logic without DPR.

V. Conclusion

This paper has compared experimentally three point-to-multipoint interconnects: ρ-P2P, μ-P2P and a full crossbar, in order to assess in details the benefits of using a DPR-based interconnect. The comparison has been performed in terms of resource needs and maximal operating frequency with respect to the number of input/output flows, their width and the number of communication configurations.

We demonstrate experimentally that ρ-P2P is a resource efficient solution if relatively large numbers of inputs/outputs are required. When the focus is set on throughput, the full crossbar is the best solution among the three. When the number of configurations is very small, which is a minor case, μ-P2P becomes the best in frequency and resources. In terms of flexibility and configuration time, ρ-P2P suffers from limitations that reduce the number of cases where it can effectively be used. However, when it comes to resources, ρ-P2P starts to gain ground when the communication complexity is above 8 inputs/outputs. After this point, ρ-P2P becomes much less resource intensive than the two other solutions and its size grows only linearly with the number of I/Os while the crossbar size grows quadratically.

VI. Acknowledgments

This work was supported by the French *Agence Nationale de la Recherche (ANR)* in the context of the High Performance embedded Computing (HPeC) project.

References

[1] L. Benini and G. De Micheli, "Networks on chips: a new soc paradigm," *computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] J. Y. Hur, S. Wong, and S. Vassiliadis, "Partially reconfigurable point-to-point interconnects in virtex-ii pro fpgas," in *International Workshop on Applied Reconfigurable Computing*. Springer, 2007, pp. 49–60.

[3] *Cyclone V Device Handbook : Volume 1: Device Interfaces and Integration*, Intel Altera, 2016, 2016.06.10.

[4] D. Koch, *Partial Reconfiguration on FPGAs: Architectures, Tools and Applications*. Springer Science & Business Media, 2012, vol. 153.

[5] L. Devaux, S. B. Sassi, S. Pillement, D. Chillet, and D. Demigny, "Flexible interconnection network for dynamically and partially reconfigurable architectures," *International Journal of Reconfigurable Computing*, vol. 2010, p. 6, 2010.

[6] C. Bobda and A. Ahmadinia, "Dynamic interconnection of reconfigurable modules on reconfigurable devices," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 443–451, 2005.

[7] S. Jovanovic, C. Tanougast, S. Weber, and C. Bobda, "Cunoc: A scalable dynamic noc for dynamically reconfigurable fpgas," in *2007 International Conference on Field Programmable Logic and Applications*. IEEE, 2007, pp. 753–756.

[8] T. Pionteck, R. Koch, and C. Albrecht, "Applying partial reconfiguration to networks-on-chips," in *2006 International Conference on Field Programmable Logic and Applications*. IEEE, 2006, pp. 1–6.

[9] P. Marquet, S. Duquennoy, S. Le Beux, S. Meftali, and J.-L. Dekeyser, "Massively parallel processing on a chip," in *Proceedings of the 4th International Conference on Computing Frontiers*, ser. CF '07. New York, NY, USA: ACM, 2007, pp. 277–286.

[10] S. Young, P. Alfke, C. Fewer, S. McMillan, B. Blodget, and D. Levi, "A high i/o reconfigurable crossbar switch," in *Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on*. IEEE, 2003, pp. 3–10.

[11] C. H. Hoo and A. Kumar, "An area-efficient partially reconfigurable crossbar switch with low reconfiguration delay," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2012, pp. 400–406.

[12] A. Morales-Villanueva and A. Gordon-Ross, "Partial region and bitstream cost models for hardware multitasking on partially reconfigurable fpgas," in *Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International*. IEEE, 2015, pp. 90–96.

[13] *Quartus Prime Standard Edition Handbook Volume 1: Design and Synthesis*, Intel Altera, 2015, 2015.11.02.