NoC based virtualized FPGA as cloud Services

Hiliwi Leake Kidane, El-Bay Bourennane
Laboratorie Le2i
Université Bourgogne Franche-Comté
21000 Dijon, France
Email:hiliwi-leake.kidane@u-bourgogne.fr, ebourenn@u-bourgogne.fr

***Abstract*—Web-based applications are increasingly demanding many computationally intensive services. On the other hand, FPGA-based hardware accelerators(HwAcc) provide good performance in accelerating computationally intensive applications. In addition, some FPGAs support a dynamic partial reconfiguration (DPR) techniques to virtualize and share the FPGA underlying hardware resources in time multiplexing during run-time to save resource and power consumption. Integrating FPGA in a cloud environment is an indispensable way to improve efficiency and provide acceleration services to demanding users. More importantly, in recent years it was proved that FPGA resources deployed in a cloud environment can be accessed with the same OpenStack software technology used to access virtual machines. However, the performance of the virtualized FPGA is highly dependent on the communication medium used to interconnect the virtualized FPGA resources and the control manager. After analyzing the possible interconnect mediums, we have selected Network-on-Chip (NoC) which support parallel communication as the efficient medium for accelerators. Consequently , we propose a NoC based virtualized FPGA as cloud Services. Two virtualized FPGA-based cloud service: Hardware Accelerator as a Service(HAaaS) and Reconfigurable Region as a Service(RRaaS) are proposed in this paper. The NoC provides layered and parallel communication between the virtualized regions of the FPGA and helps them to communicate their status and exchange data through the routers connected to them. A 2x2-mesh NoC based reconfigurable accelerators for image analysis and matrix computation are implemented and tested showing a promising result for more scalable systems in cloud computing.**

***Index Terms*—Cloud Computing; Virtualized FPGA; Hardware accelerators; Network-on-Chip;**

I. Introduction

Web based applications are increasingly demanding many services which are highly resource consuming that can not be performed locally [1]. Cloud computing minimizes these problems by offering software, platform and infrastructure as a service. The cloud makes it possible for a user to access information from anywhere at any time [2]. It uses billing mechanisms to use these resources on the basis of their consumption, allowing on-demand model: pay-per-use [3].

On the other hand, Field programmable Gate Array (FPGA)-based hardware accelerators(HwAcc) provide good performance in accelerating computationally intensive applications [4] like multimedia image analysis. In addition, some FPGAs support a dynamic partial reconfiguration (DPR) techniques to virtualize and share the FPGA underlying hardware resources in time multiplexing during run-time to save resource and power consumption.

Consequently, cloud provider can improve their computing performance and provide accelerating service by integrating virtualized FPGA in a cloud environment. More importantly, in recent years, it has been demonstrated that FPGA resources deployed in a cloud environment can be accessed with the same Openstack software [5] technology used to access virtual machines [6], [7], [8]. OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter [9].

However, communication issues could hamper the proper operation of such method if proper communication medium is not proposed. When the number of processors in a given FPGAs increases, the point to point connection using the ordinary wire or bus based communication is not efficient and reliable. On the other hand, the Networks-on-Chip (NoCs) have recently emerged as a promising concept to support communication on Multi-Processor SoCs [10] due to their scalable and layered architecture. Thus, in this paper we propose a method that takes best of both worlds: the improved communication features of NoCs with the adaptability on demand provided by dynamically reconfigurable systems in order to integrate the virtualized FPGA and provide its resources as a services.

II. Background

Before proceeding to the main contributions of this paper, we will first provide some basic concepts and terminology.

A. Cloud Computing

Cloud computing is the provision of computing resource from remote on-demand [16]. The cloud services are mainly divide into three categories [2]: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) as shown in Fig. 1.

There are some essential characteristics to consider during any cloud service proposal [2] [16] [3]. The first characteristic is on-demand self service or service without human interaction. The second characteristic refers to the availability of the service over the network and accessibility by standard client platforms. The third characteristic is related to the resource pooling or parallelism. In other words, the service must serve multiple clients at the same time. The other important characteristic is rapid elasticity of the service. When clients want to extend the resource they are using, the service should allow them to scale it dynamically. The final and most

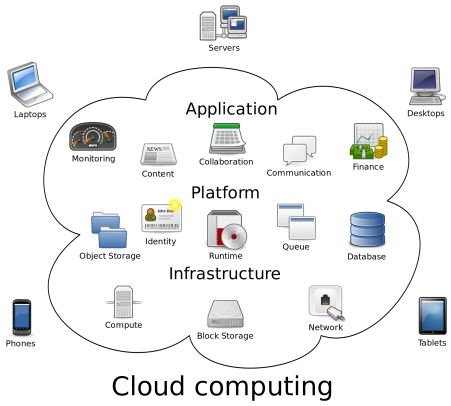

Fig. 1. Cloud Computing (source: wikipedia)

important characteristic is the service has to be measurable and the control has to measure and report service usage for payment.

B. Network-on-Chip (NoC)

1) building blocks: All NoCs have three fundamental building blocks, namely, switches (also called routers), Network Interfaces (NIs) and links [12] as shown in Fig. 2. A Router is responsible for the routing or directing of data based on the protocol defined at that moment. The router contains an arbiter, a buffer and an input-output to connect ports. Links are the channels that connect router to router or router to NI. NI is an intermediate between the router and the processing element (PE) connected to the router. The network interface (NI) is responsible for packetization and depacketization of data traffic, in addition to the conventional interfacing [13].

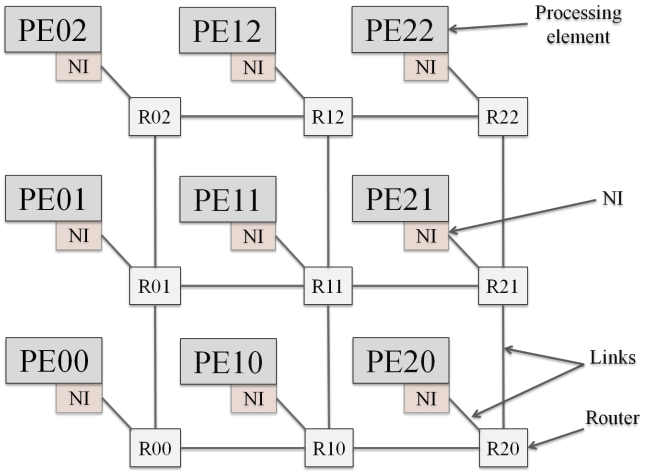

Fig. 2. Network-on-Chip (NoC) based communications of PEs

2) Topology: refers to the interconnection between NoC building blocks. The topology can be categorized as regular and irregular based on the distribution of the routers in the network. The regular topology also contains different varieties like mesh, mesh tours, ring, fat tree. The performance of NoC based communication strongly correlates with the topology selected for implementation [14].

3) Switching: determines how the data is transferring from one node to the other. The two common switching techniques are packet switching and circuit switching. Circuit switching, the link is used in a spatial division approach until the data transmission is completed. Whereas in the packet switching the links are used in a time division approached. Different packets of the same data can follow different path [15].

4) Routing: refers to the algorithm in the routers which selects an output port for the packet coming through its input port. There are different varieties of routing algorithms which give different results [13].

5) Flow Control: determines how network resources, such as channel bandwidth, buffer capacity, and control state, are allocated to a packet traversing the network [13]. An efficient design of flow control is crucial in order to get good performances.

C. Dynamic Partial Reconfiguration

Dynamic partial reconfiguration (DPR) is a technique that enables to dynamically modify preselected area of the FPGA at run-time and on demand. The DPR is not supported by all FPGAs. The Atmel FPGA and Xilinx FPGAs are some of the few FPGAs in the market allowing DPR. Figure 3 shows the basic premise of Partial Reconfiguration (PR). The reconfigurable region represented by ”A” can be used to configure and run different modes of ”A” in time multiplexing.

The general PR flow as stated in the Xilinx PR flow [11] is as follows. The logic in the FPGA design is divided into static logic and reconfigurable logics/modules. The static logic does not change its functionality during reconfiguration and it contains an embedded processor, an internal configuration access port (ICAP), DSP and other circuitry. The embedded processor runs the configuration management software which controls system transition from one configuration to another depending upon the adaptation conditions set by the application. The ICAP is used to load/transmit the partial bitstreams into the FPGA configuration infrastructure.

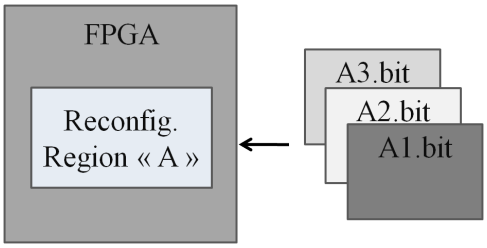

Fig. 3. Basics of Partial reconfiguration

The DPR process starts by implementing the static or top level HDL module which includes a black box for the partially reconfigurable modules and then synthesized to generate the top level netlist. The HDL of the different modes of the partially reconfigurable modules should be implemented

and synthesized separately to generate independent netlists. Subsequently, both the top level and partial reconfigurable module Netlists are imported to the PlanAhead to floorplan the placement and mapping of the Reconfigurable regions and reconfigurable modules. Placement restriction must be associated to reconfigurable IPs during DPR design to define their size, shape and position.

III. Previous Works

The use of virualized FPGA accelerator in datacenters for better performance and flexibility is presented in [17], [18], [19]. Authors in [20] presented a prototype for integrating virtualized FPGA accelerators in the cloud using partial reconfiguration and virtual communication. Similarly, [21] proposes a resource virtualization solution based on run-time partial reconfiguration to share reconfigurable resources among the underutilized microprocessors. In addition, [6], [7], [8] presented that FPGA resources deployed in a cloud environment can be accessed with the same Openstack software technology used to access virtual machines.

Bharathi and Neelamegam [1] propose a Reconfigurable Framework for Cloud Computing Architecture with three layer called service usage, service provider and service developer. If the demanding service from a user is not available, service usage module requests service developer module through service provider module to develop a new or modify existing service to satisfy the customer application need on demand. Reconfiguration takes place if the service is not available in the hardware. Reconfiguration bitstreams are generated in service developer module and transferred to the hardware with CPU control.

Knodel and Spallek [22] present a cloud service models and cloud hypervisor called RC3E, which integrates virtualized-FPGA based hardware accelerators into a cloud environment. The authors defined three types of service called RSaas, RAaaS and BAaaS. The RSaas service provides full access to reconfigurable resources for the user. The user can allocate a complete physical FPGA with own implemented hardware. In RAaaS, the FPGA is used as a simple accelerator and only the vFPGAs or virtual reconfigurable regions of different size are visible and accessible by the user. In the last service, BAaaS, only available applications and services are visible to the user. These applications and service use the virtual reconfigurable regions of the FPGA(vFPGA) in the background to accelerate specific applications.

Even though most of the authors provide detailed information about the advantages of integration virtualized FPGA for cloud computing, they all failed to give attention to the complexity of communication requirement for data as well as control signal communication.

IV. NoC based virtualized FPGA as cloud Services

In this paper, we have proposed two NoC based virtualized FPGA as service models for cloud computing called Hardware Accelerator as a Service(HAaaS) and Reconfigurable Regions as a Service(RRaaS). The NoC will help to easily access the reconfigurable IPs and send signal to the control up on completion of tasks.

The proposed virtualized FPGA (vFPGAs) as cloud services are based on 2D-mesh Hermes NoC [24] architecture. Each vFPGAs is connected to the nearest router through network interface(NI). The internal structure of the NI depends on the ports of the vFPGAs connected to it. The NI will also serve as an isolation during reconfiguration of the vFPGAs connected to it.

The architecture has three layers: the virualized FPGA or reconfigurable regions, the hypervisor which contains the static region of the FPGA and the application layer which gives interface to the external user as shown in figure 4. The Hypervisor is responsible for the management of reconfiguration and resources and many virualized machines are installed over a single hypervisor.

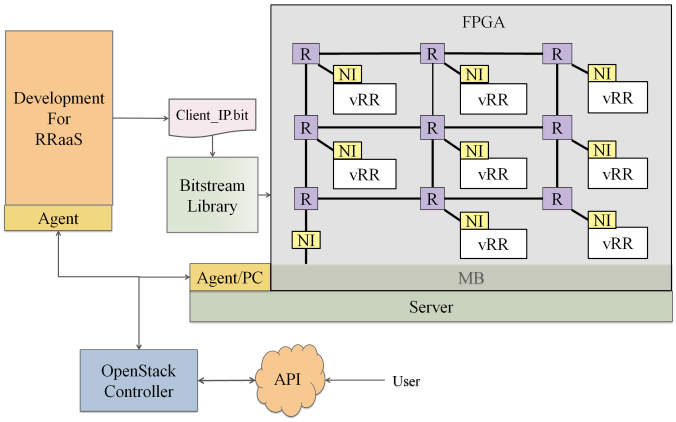

Fig. 4. Architecture of the NoC based virtualized FPGA as cloud Services

A. Hardware Accelerator as a Service(HAaaS)

In this service model, the user does not have direct access to the virtualized FPGA. When the user requests any accelerator, the intermediate control manager checks availability of the requested accelerator and if it exists, it establishes connection between accelerator and the user in the host machine. If not, the control manager demands to reconfigure the requested accelerator from the existing bitstream library as stated in the flow diagram in Figure 5.

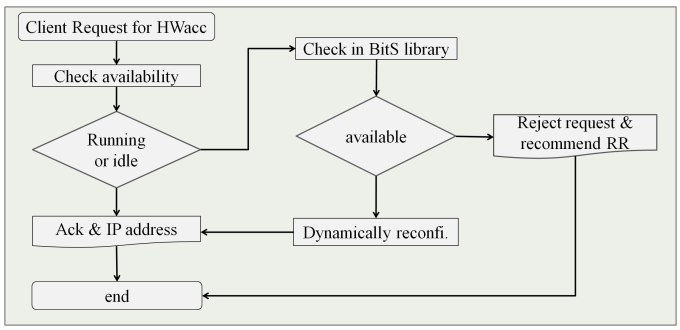

Fig. 5. Flow diagram of HAaaS

In case, the requested accelerator does not exist in the library at all, the control manager has to reject the request and recommends the user to use the other service RRaaS.

B. Reconfigurable Regions as a Service(RRaaS)

This services can be divided into two stages as development and provision. First, the user get access to select one of the available virtualized RRs based on their top-level architecture and capacity in terms of resourses. Next, the control manager send back the HDL template file of the selected vRR to the user to implement the detail design. Later, the user will send back the completed HDL implementation via Ethernet. Finally, the control manager in the service provider will send to the development to synthesize, and if no error, to place it in optimal vRRs for bitstream generation. This is the development stage. In the second stage, if the user wants to run his user IP, the generated bitstream will be forwarded to the bitstream library. If not the user gets only the generated bitstream file. The detailes of the flow is given in Figure 6.

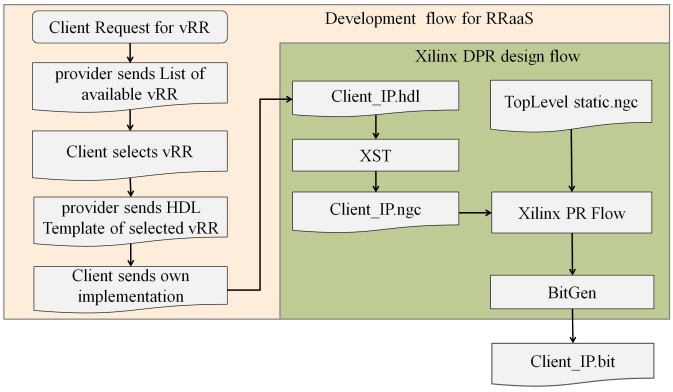

Fig. 6. Flow diagram of RRaaS

To optimize the resource utilization, the size of the virtualized regions are grouped into three as small medium and large. That means, and HDL templete file of the RR region is suitable to the three groups. This will help the control manager to place the user IP in optimal region.

C. Hypervisor

Normally the provision of virtualized machines has three layers; the application layer, hypervisor and the physical layer. The application layer in this case corresponds to the Openstack API-agent interface software. The hypervisor in virtualized FPGA corresponds to the static region or module of the top level structure of the FPGA. Then, the virtualized reconfigurable regions will be instantiated and controlled by the hypervisor. In other words, the hypervisor is the one that manage the virtualized FPGA resources.

V. Experimental results

To test the proposed algorithm, we have implemented image analysis and mathematical manipulation modules and defined three reconfigurable regions for virtualization in the FPGA as shown in Figure 7.. A 2x2-mesh NoC generated from the Atlas NoC generator [24] is used as communication medium. The first router is directely connected to the control/hypervisor and the rest three routers to the virtualized reconfigurable regions. The second router is connected vRR1 where vRR1 is designed for scalar manipulation and includes three modes; addition, subtraction and multiplication. The third router is also connected to vRR2 where this virtulized region is designed for accelerating image processing modes: median filtering, Gray scale and contrasting. Finally, the last router is connected to vRR3 where vRR3 is dedicated for accelerating 8x8 matrix addition and subtraction.

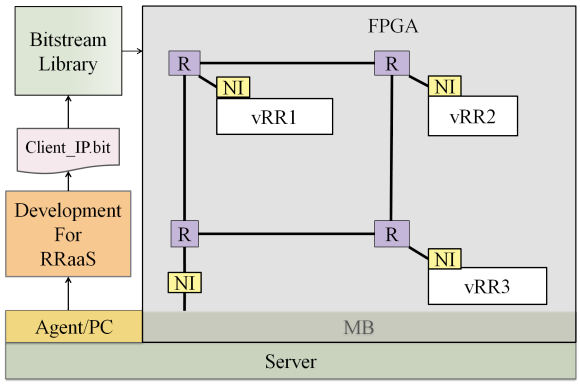

Fig. 7. 2x2 NoC based reconfigurable IPs

A performance evaluation in terms of resource utilizationa and reconfiguration time for the virtualized reconfigurable regions is given in Table-1. The result shows that that the size of the bitstream is too small and generating as much as possible of reconfigurable modules will not cost memory. The reconfiguration time is also to small which is applicable for real time.

TABLE I
Performance evaluation

DPR Module	Resources Utilization				Bitstream size	config. time
	LUT	FF	DSP	BRAM		
vRR1	2013	1208	12	1	17 KB	0.21 ms
vRR2	9193	7470	40	1	34 KB	0.43 ms
vRR3	5419	6121	40	8	29 KB	0.39 ms

TABLE II
Resource utilization of the 2x2 NoC

LUT	FF	DSP	BRAM
6117	2958	3047	6

As the services provided by a cloud computing must be measurable, there should be continuous status communication between the virtualized reconfigurable regions and the hypervisor which control and manages the resources utilization. It also helps to replace idle accelerators by the blank bitstreams so that power can be saved. It is observed that loading a blank bitstream during idle time saves about 35% of the power

consumption when a real accelerator is loaded and is not performing anything. The NoC based communication helps the structure to have parallel communication and perform both data and control signal communication at the same time.

VI. Conclusions

We have implemented a NoC based virtualized FPGA and tested locally to share the FPGA resources on cloud context. Integration of hardware accelerators in the cloud environment helps to provide FPGA resource in a cheap price and improve the utilization of the FPGA. Similarly, the performance of a cloud environment will be maximized by integrating hardware accelerator FPGA.

The NoC will provide a flixible and scalable parallel communication architecture to the virtualized FPGA resources so that both data and control signal communication can be performed in parallel. We have presented the advantage of integrating a NoC based reconfigurable accelerators in the cloud computing. Two possible virtualized FPGA cloud based service models are proposed in this paper. The first service model, HAaaS, helps the cloud provider to give accelerating services. Whereas the second service model, RRaaS, provides both development and accelerating service. As a future work, the NoC will be extended into dynamically reconfigurable and then it will be deplyed into a server to test it via API.

References

[1] N. Bharathi and P. Neelamegam, "A reconfigurable framework for cloud computing architecture," *Artificial Intelligence*, vol. 6, pp. 117–120, 2013.

[2] A. Huth and J. Cebula, "The Basics of Cloud Computing," *United States Computer Emergency Readiness Team*, pp. 1–4, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124059320000074

[3] L. Schubert, K. G. Jeffery, and B. NeideckerLutz, "The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010:-expert Group Report," Tech. Rep. European Commission, Information Society and Media, 2010.

[4] T. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. Kindratenko, and D. Buell, "The promise of high-performance reconfigurable computing," *Computer*, vol. 41, no. 2, pp. 69–76, 2008.

[5] Openstack. [Online]. Available: http://www.openstack.org/software/

[6] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 13–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665678

[7] S. Byma, J. Steffan, H. Bannazadeh, A. Leon-Garcia, and P. Chow, "Fpgas in the cloud: Booting virtualized hardware accelerators with openstack," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, May 2014, pp. 109–116.

[8] J. Dondo Gazzano, F. Sanchez Molina, F. Rincon, and J. C. López, "Integrating reconfigurable hardware-based grid for high performance computing," *The Scientific World Journal*, vol. 2015, 2015.

[9] Openstack, *OpenStack Operations Guide*. [Online]. Available: http://docs.openstack.org/ops-guide/index.html

[10] R. Dafali, J.-P. Diguet, and M. Sevaux, "Key research issues for reconfigurable network-on-chip," in *Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on*, Dec 2008, pp. 181–186.

[11] Xilinx, "Partial Reconfiguration User Guide," vol. 702, 2013.

[12] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, and G. D. Micheli, "Network-on-chip design and synthesis outlook," *Integration, the {VLSI} Journal*, vol. 41, no. 3, pp. 340 – 359, 2008.

[13] A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (noc) architectures & contributions," *Journal of engineering, Computing and Architecture*, vol. 3, no. 1, pp. 21–27, 2009.

[14] P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. De Micheli, "Design, synthesis, and test of networks on chips," *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 404–413, Sept 2005.

[15] É. Cota, A. Morais Amory, and M. Soares Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*. Springer, 2012.

[16] F. Alshuwaier, A. Alshwaier, and A. Areshey, "Applications of cloud computing in education," in *Computing and Networking Technology (ICCNT), 2012 8th International Conference on*, Aug 2012, pp. 26–33.

[17] P. Francisco, "The Netezza data appliance architecture: A platform for high performance data warehousing and analytics," *IBM Redbooks*, 2011.

[18] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, June 2014, pp. 13–24.

[19] J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, and S. Jiang, "SDA : Software-Defined Accelerator for Large- Scale DNN Systems," in *HOT CHIPS*, 2014, pp. 1–23.

[20] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized fpga accelerators for efficient cloud computing," in *Cloud Computing Technology and Science (CloudCom), 2015 IEEE 7th International Conference on*, Nov 2015, pp. 430–435.

[21] E. El-Araby, I. Gonzalez, and T. El-Ghazawi, "Virtualizing and sharing reconfigurable resources in high-performance reconfigurable computing systems," in *High-Performance Reconfigurable Computing Technology and Applications, 2008. HPRCTA 2008. Second International Workshop on*, Nov 2008, pp. 1–8.

[22] O. Knodel and R. Spallek, "Computing framework for dynamic integration of reconfigurable resources in a cloud," in *Digital System Design (DSD), 2015 Euromicro Conference on*, Aug 2015, pp. 337–344.

[23] H. L. Kidane, E. Bourennane, and G. Ochoa-Ruiz, "Noc based virtualized accelerators for cloud computing," in *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2016 IEEE 10th International Symposium on*, Sept 2016, pp. 133–137.

[24] F. Moraes, N. Calazans, A. Mello, L. Mller, and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," *Integration, the {VLSI} Journal*, vol. 38, no. 1, pp. 69 – 93, 2004.