Synchronizing reconfiguration of coherent functions on disaggregated FPGA resources

Qianqiao Chen[†*], Vaibhawa Mishra[*], Jose Nunez-Yanez[†], Georgios Zervas[*]
[*]University College London, London, UK
{qianqiao.chen, vaibhawa.mishra, georgios.zervas}@ucl.ac.uk
[†]University of Bristol, Bristol, UK
{qianqiao.chen, j.l.nunez-yanez}@bristol.ac.uk

***Abstract*—With the evolution of cloud computing, FPGAs are involved in the data centers thanks to their high performance and logic reconfigurable features. To efficiently make use of data center resources, recent rack scale architecture tends to disaggregate data center resources. This paper proposes the Synchronizing Network wide Function Reconfiguration (*SNFR*) protocol that aims to synchronize the reconfiguration of coherent functions on disaggregated FPGA resources deployed across the network. The associated protocol processor is implemented. The synchronized reconfiguration is captured by the Xilinx debug core and network traffic analyzer. The experimental section shows that the protocol processor can support maximum 9 Gbps traffic and introduces additional latency ranged from from 0.1 μs to 0.21μs.**

I. Introduction

Cloud computing has become a significant part of our daily lives, thanks to its Internet-scale services, such as web search engine and e-mail. Data centers have been established to support the rapidly growing cloud computing services. Common data center infrastructures include a number of servers interconnected by a network. To efficiently make use of data center resources, recent rack scale architectures propose to disaggregate computing resources [1]. The resource disaggregation suggests that each server main board contains only a unique type of computing resource. When interconnected by the network, these servers along with its associated resources can be as a single logical hardware platform to support cloud applications.

Due to the feature of parallel processing and reconfiguration, FPGA usually performs as a programmable accelerator alongside the processor. Recent FPGA technologies such as high-level synthesis and partial reconfiguration require more FPGA resources in the high performance computing. Frameworks have been proposed to manage the FPGA resources in the traditional data center [2]. As a next step, the FPGA resources can also be directly attached to the network and become disaggregated resources in data centers. However, real-time reconfiguration of coherent functions; functions that are highly related to each other, for example, encoder/decoder and compressor/decompressor, on disaggregated FPGA resources becomes a big challenge, because propagating reconfiguration requests to multiple FPGAs through network is hard to synchronize due to the unpredictable network latency.

This paper proposes a method to enable synchronized reconfiguration of coherent functions on network-attached disaggregated FPGA resources. The main contributions of this paper are:

- A network protocol named Synchronization of Network wide Function Reconfiguration (*SNFR*) is proposed. The *SNFR* packets are able to carry information for the reconfiguration of multiple network-attached FPGAs.
- The processor for the *SNFR* protocol is implemented. It is able to perform seamless function reconfiguration on disaggregated FPGA resources.

A reconfiguration process over 10 Gbps Ethernet traffic has been demonstrated. The reconfiguration is captured by Xilinx debug core and traffic analyzer. The influence of the proposed *SNFR* protocol processor on the latency is measured. The protocol processor is able to supports maximum total of 9 Gbps traffic (out of which 8.1 Gbps correspond to regular data flows and 0.9 Gbps represent *SNFR* flows).

The rest of the paper is organized as follows: Section II introduces the related work about the disaggregation and reconfiguration of FPGA resources in data centers. Section III motivates the synchronization of coherent function reconfiguration on disaggregated FPGA resources. The proposed *SNFR* protocol and the associated protocol processor are introduced in section IV. The proposed reconfiguration method is demonstrated in section V. Section VI concludes this paper.

II. BACKGROUND

With the evolution of cloud computing, FPGAs get more involved in high performance computing due to its high performance and logic configurable architecture. Researchers in [3] [4] propose to integrate the FPGA resource management with existing data center resource manager. Moreover, high level synthesis is suggested in [5] to bridge the FPGA design with software data center controller and further reduce the deployment and design complexity. FPGA resource disaggregation has also been suggested in [6]. By attaching FPGAs directly to the network, J. Weerasinghe et. al enables the deployment of large-scale independent FPGA resources in the data center [6].

Furthermore, researchers have reported methods to manage the reconfiguration process of the FPGA resources. For example, L. Pezzarossa et. al proposed to manage the FPGA in a

time-division-multiplexing manner [7]. Interfaces of reconfigurable regions are controlled by a predefined scheduler. The reconfiguration can happen in high frequency manner in this approach, because the time slots when deploying functions are very short (thousands of clock cycles). However, as this approach is developed for single chip reconfiguration, it is designed based on a unique global clock and time-predictable FPGA components. However, in the data center environment, synchronized global clock can result in additional cost and predicting the timing of every component will be difficult. On the other hand, another approach to manage the reconfiguration process of FPGA resources through resource controller of data centers has also been investigated. OpenStack [8] is adopted to schedule the reconfiguration of FPGA resources. FPGA devices are virtualized and partitioned into many partially reconfigurable regions and the reconfiguration process is managed by a kernel virtual machine [9] [10] [11]. The resource allocation of these approaches cannot meet the dynamic requirement of reconfigurable computing, as the reconfiguration process takes longer time and introduces downtime.

A protocol processor that synchronizes the reconfiguration of coherent functions across multiple network-attached disaggregated FPGA resources is proposed in this paper. It ensures seamless function reconfiguration on network flows without downtime.

III. Motivation

This section motivates the synchronization of coherent function reconfiguration on disaggregated FPGA resources. The idea of the proposed *SNFR* protocol is also briefly introduced.

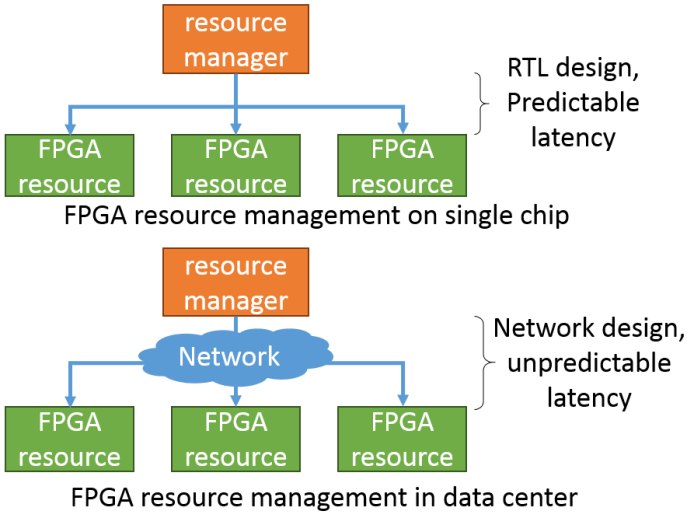

Fig. 1. Reconfiguration of FPGA functions on single chip and in data center.

The difference between the reconfiguration on single chip and across disaggregated resources throughout the data center is shown in Fig.1. The reconfiguration process on single chip is triggered locally through duplicated signals in RTL design. The time when reconfiguration request arrives at the FPGA resource is accurately counted in clock cycles and can be predicted by simulation tools. However, in the data center environment, the reconfiguration request is encapsulated in the packet and is transferred through the data center network. The latency of the request is hard to predict because it might be affected by the network congestion and routing strategy.

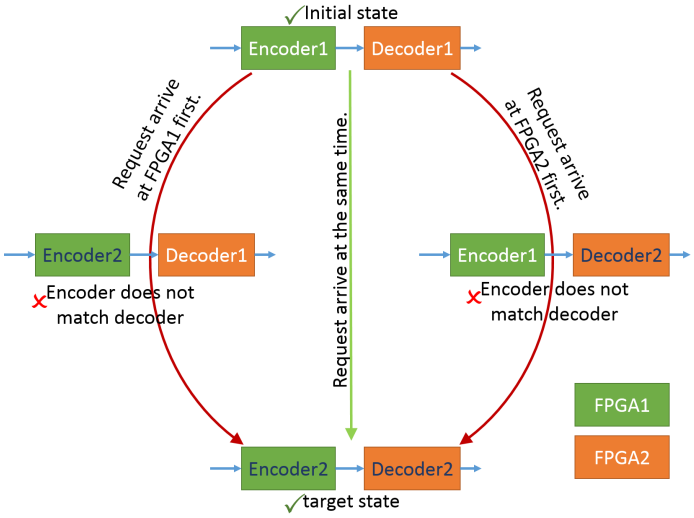

Fig. 2. An example for network wide coherent function reconfiguration.

The unknown latency in the data center environment can result in downtime when coherent functions are being reconfigured. An example that the unpredictable latency of the reconfiguration request leads to unexpected failure state is shown in Fig.2. The hardware platform is composed of two network attached FPGAs. The network data flow is firstly encoded in $FPGA_1$. It is then forwarded over network switches to $FPGA_2$ and decoded. The reconfiguration aims to switch from the initial $encoder_1$/$decoder_1$ to $encoder_2$/$decoder_2$. Due to the unknown latency of the reconfiguration request, the system may jump into failure state where the encoder and decoder of the data flow do not match.

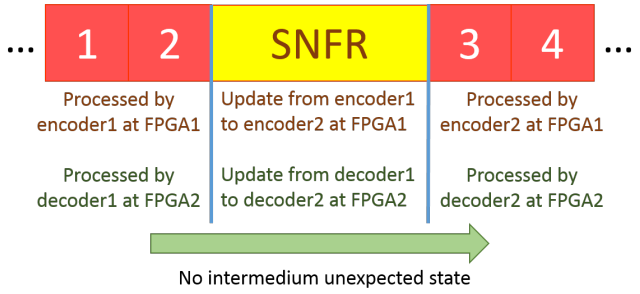

Fig. 3. The solution to synchronize the reconfiguration of coherent FPGA functions on disaggregated FPGA resources.

A method to synchronize the reconfiguration of coherent functions on disaggregated FPGA resources in data center environment is proposed in this paper as shown in Fig.3. A *SNFR* packet is inserted in the data flow by the source node/server. This *SNFR* packet carries the information of reconfiguration

for all the FPGAs on the path of the flow. The *SNFR* packet will go through all the FPGAs in the path together with the data flow. When a *SNFR* packet reaches a FPGA, the data flow is buffered. The information of reconfiguration for the current FPGA is extracted from the *SNFR* packet, so that the function on the current FPGA is reconfigured accordingly. Then the *SNFR* packet and the data flow are released and are forwarded to the next FPGA after processing by the on-chip functions. As shown in Fig.3, coherent functions on different FPGAs are reconfigured at the same location of the data flow (The location of the inserted *SNFR* packet).

IV. IMPLEMENTATION

The implementation of the system to synchronize the function reconfiguration is shown in this section. Especially, the proposed *SNFR* network protocol and its associated protocol processor is introduced.

A. Overall architecture of the reconfigurable FPGA system

A reconfigurable network on chip platform is introduced in [12]. As shown in Fig.4.(a) right, it is composed of a 10 Gbps Ethernet interface, several partial reconfigurable regions, an on chip interconnect and a *SNFR* protocol processor. Network flows can be forwarded from 10 Gbps Ethernet interface to reconfigurable regions to enable reconfigurable pipelined data stream processing/acceleration. To change the connection between reconfigurable regions at run time, the port map of the on chip interconnect is dynamic and can be updated through memory mapped AXI interface. The *SNFR* protocol processor is located between the 10 Gbps Ethernet interface and the on chip interconnect. It is able to identify the inserted *SNFR* packets from regular packets, buffer/release the traffic and update the port map of the interconnect according to the information of reconfiguration provided by the *SNFR* packets.

B. The proposed protocol for coherent function reconfiguration

The detailed architecture of the protocol processor is shown in Fig.4.(a) left. It is composed of FIFO, FIFO controller, protocol processing unit and master AXI4 interface. Data flows received from 10 Gbps Ethernet interface are sent to the FIFO directly. When the FIFO controller detects a *SNFR* packet, it acknowledges the FIFO to start buffering the traffic and it also acknowledges the protocol processing unit to start extracting information of reconfiguration from the *SNFR* packet. The address and data information are extracted from the *SNFR* packets by the protocol processing unit. The extracted data and address information are then transferred to the master AXI4 interface to handle the memory mapped AXI4 handshake and update the routing table of the on chip interconnect. When the AXI4 master receives successful responds for all the transfers, the protocol processing unit notifies the FIFO controller to release the traffic.

The proposed packet data structure to support the synchronized reconfiguration is shown in Fig.4.(b). The *SNFR* packet is encapsulated in the IP packet and is identified by the protocol field of the IP protocol. The definition of each field of the *SNFR* protocol is listed below:

- Offset: 64 bits. This field marks the starting point where the current protocol processing unit should extract data and address information.
- ADD: 32 bits. If it is not 0xFFFF_FFFF, this value should be transferred to the memory-mapped AXI interface as an

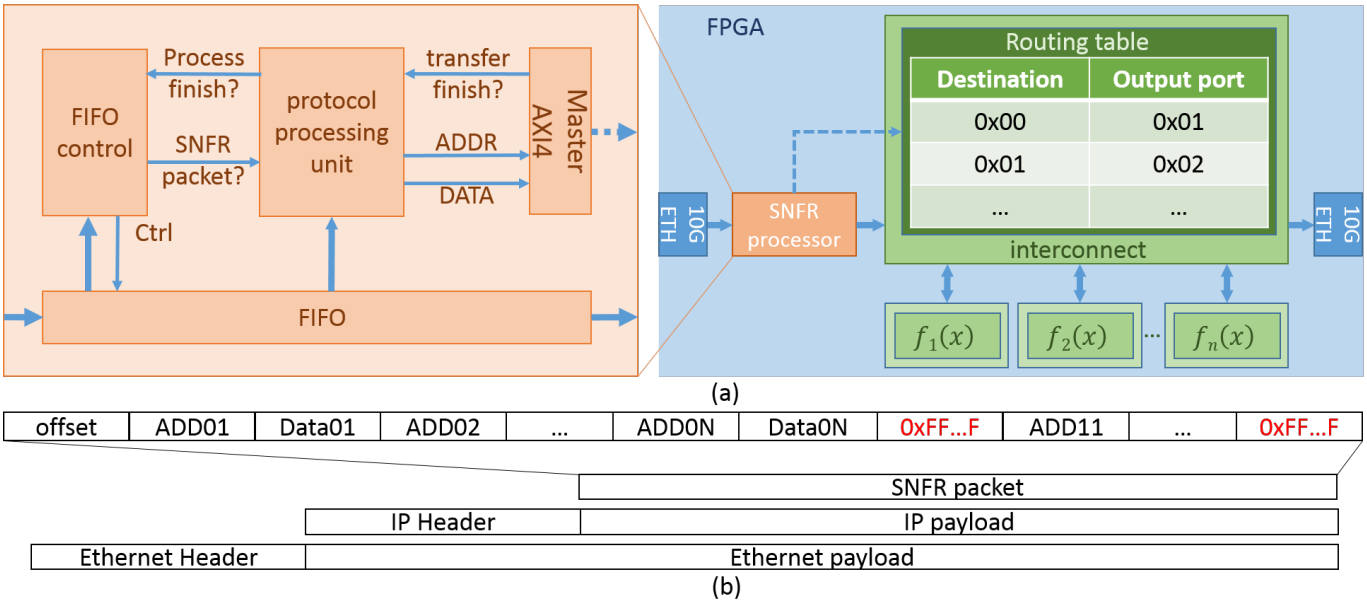

Fig. 4. The proposed system that synchronize the reconfiguration of coherent disaggregated FPGA functions. a) The overall system and the detailed architecture of the protocol processor. b) the proposed protocol to carry the reconfiguration information

address. If it is 0xFFFF_FFFF, the protocol processing unit should stop extracting data and address.

- DATA: 32 bits. The data that should be transferred to the memory-mapped AXI interface.

Fig. 5. The brief description of the behavior of the protocol processing unit. The start location depends on the value of the offset. When the process is finished, the current processing unit will update the offset value to make it point at the start location for the next FPGA.

The behavior of the protocol processing unit is shown in Fig.5. When a *SNFR* packet is received, the protocol processing unit will extract the value of address and data from the location pointed by the offset. The end of the extraction is marked by a specific address value 0xFFFF_FFFF. When the extraction of address and data is finished, the processing unit should update the offset value to the location of next pair of address and data. Then the *SNFR* packet will be sent in the received order followed by the regular packets to the next FPGA. The new offset will be checked by the next FPGA, and the information for the previous FPGA can be skipped. It should be pointed out that one *SNFR* packet carries the reconfiguration information (address and data) for all the FPGAs in the path. The address field with value 0xFFFF_FFFF cut the packet into many segments. Each segment is for the reconfiguration of one FPGA in the path.

The detailed state machine of the protocol-processing unit is shown in Fig.6. When a *SNFR* packet is received and the traffic is buffered, the processing unit first checks the offset of the *SNFR* packet. Then it sets the read address of the BRAM to the offset value to read the first pair of address and data. The value of the address is checked: if it is not 0xFFFF_FFFF, then the read address will be incremented to read the next pair of address and data. If the address is 0xFFFF_FFFF, the processing unit then updates the offset field of the *SNFR* packet to make it point to the address and data for the next FPGA. The *SNFR* packet and the traffic is released in the next state, so the *SNFR* packet with the new offset can be sent to the next FPGA in the path.

V. Experiment and Result

This section describes the experiment conducted to demonstrate the synchronized reconfiguration on disaggregated FPGA resources. The reconfiguration of the network is captured by the FPGA Xilinx debug core and the traffic analyzer. The influence of *SNFR* packets on the latency of regular data flows is also measured. The FPGA platform in the experiment is the ZCU102 Ultrascale+ MPSoC with part number xczu9eg-ffvb1156-2-i-es2. The resource utilization of the protocol processor is shown in table I.

TABLE I
THE RESOURCE UTILIZATION OF THE PROTOCOL PROCESSOR

Component	Flip Flop	LUT	BRAM
FIFO	202	1059	6
FIFO control	174	568	0
Protocol processing unit	305	358	0
AXI master	286	148	1
Total	967	2133	7

A. Demonstration of the synchronized reconfiguration

The experimental set up and the reconfiguration process are shown in fig.8. Two FPGAs are set up and are connected

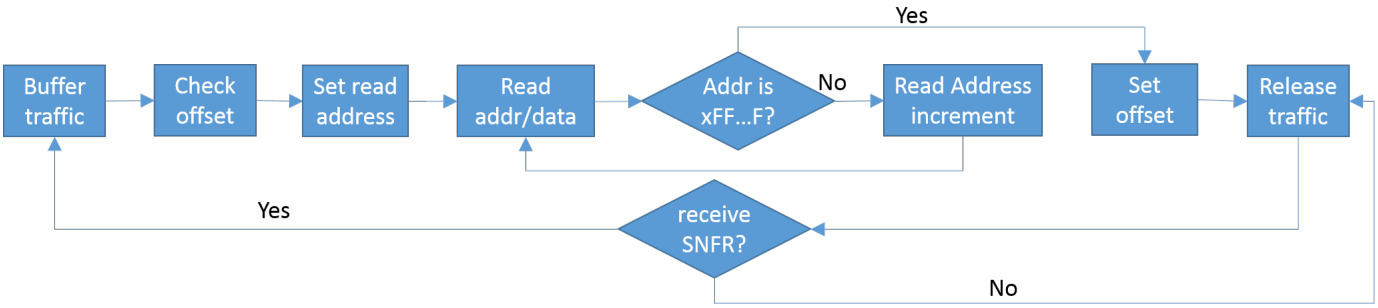

Fig. 6. The sate machine of the protocol processing unit.

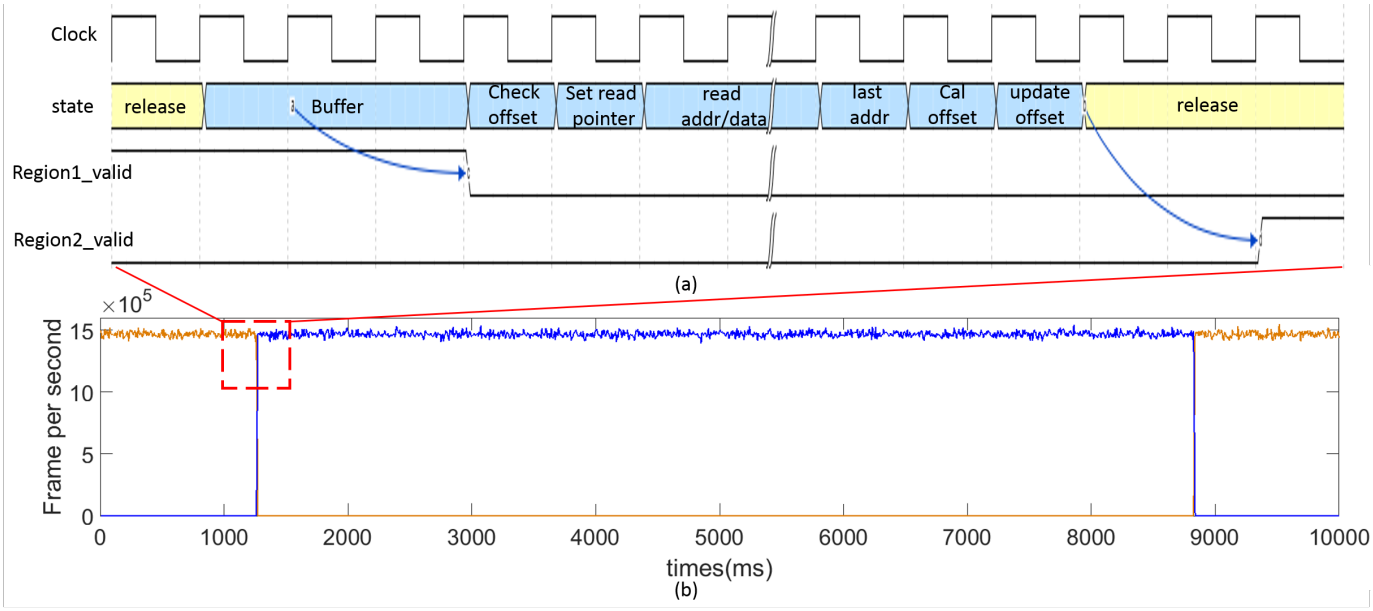

Fig. 7. The captured reconfiguration process. a) The state machine of the protocol processing unit captured by Xilinx debug core. b) The reconfiguration of output port monitored by the external traffic analyzer.

through 10 Gbps Ethernet interfaces. On each FPGA, a system is set up with an on chip interconnect, two reconfigurable regions ($f_1(x)$ and $f_2(x)$) and the protocol processor. Packet counters are deployed in each reconfigurable region to record the number of packets received by this region. The traffic is generated and analyzed by the traffic analyzer. A 9 Gbps traffic with random packet size is generated to represent the regular data flow in the data center network. Every 5 seconds, one *SNFR* packet is generated. The generated *SNFR* packets makes the data flow to continuously switch between the following two states:

- $f_1(x)$ at FPGA$_1$, $f_4(x)$ at FPGA$_2$, FPGA$_2$ forward traffic to output$_1$.

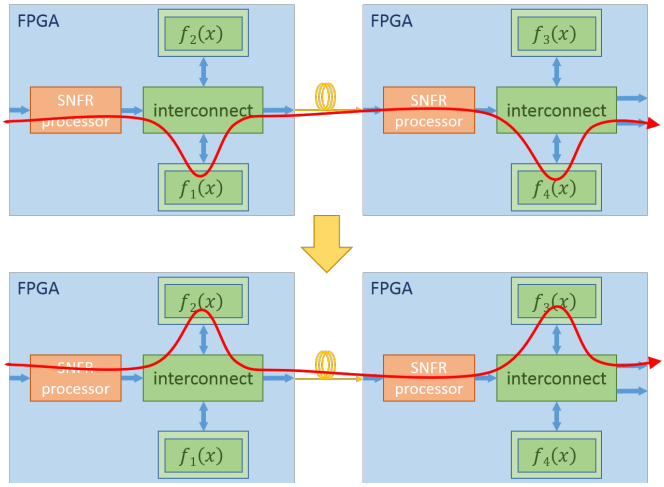

Fig. 8. The demonstrated reconfiguration process.

- $f_2(x)$ at FPGA$_1$, $f_3(x)$ at FPGA$_2$, FPGA$_2$ forward traffic to output$_2$.

We use the Xilinx debug core to monitor the state machine of the processing unit, and the valid signals of both regions as shown in Fig.7.(a). The reconfiguration process starts when a *SNFR* packet is received. To ensure data loss free operation, the *SNFR* packet is only processed when the traffic is completely buffered. We also use the traffic analyzer to monitor the output of FPGA$_2$ as shown in Fig.7.(b). The traffic is detected at output$_1$ in the initial state. After reconfiguration, the traffic from output$_1$ cannot be detected and traffic from output$_2$ is received instead.

B. Influence from the SNFR processor

The influence of the proposed protocol on the latency of the regular data flows is also measured as shown in Fig.9. To reconfigure disaggregated FPGA resources, *SNFR* packets should be inserted into the regular data flows. In this experiment, we insert the *SNFR* packets with different rates and measure the data rate vs latency chart to analyze the influence of the *SNFR* packets. It should be pointed out that the *SNFR* packet is responsible to change the state of the network, so higher *SNFR* packet rate means the network reconfiguration take places at higher frequency. As shown in Fig.9, the *SNFR* packet will only influence the traffic when the data rate of regular data flows is higher than 6 Gbps. The data rate can reach up to 0.9 Gbps, so it is able to reconfigure the FPGA resource at high rates. The protocol processor can also support high throughput. It is able to work at 9 Gbps data rate (8.1 Gbps regular data flows plus 0.9 Gbps *SNFR* flows).

The relationship between data rate of *SNFR* flows and how dynamic the network is can be represent in equation 1. The parameter is described as follows:

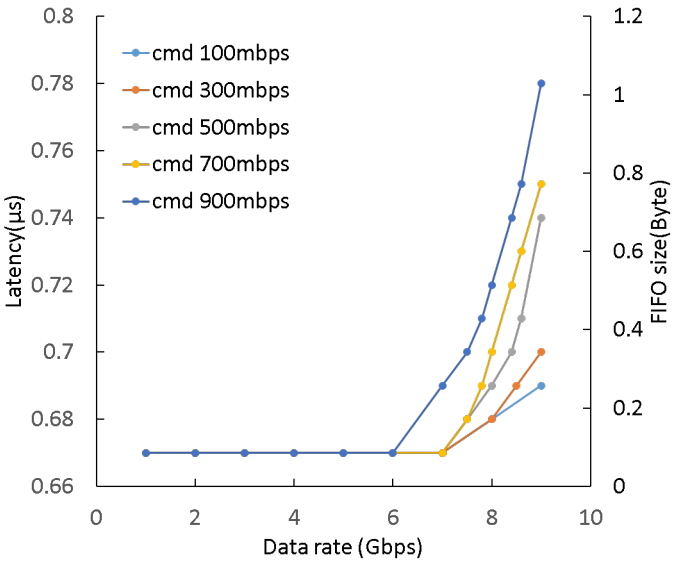

Fig. 9. The latency influence of the protocol processing.

- r: The data rate of *SNFR* flows.
- p: Every time unit, the total number of times to change the map of ports of the on-chip interconnect on all the FPGAs in the path.
- H: The size of Header.
- FPS: Frame per second.

$$r = 8 \times p + FPS \times H \tag{1}$$

The term $8 \times p$ represents the actual size of data in the *SNFR* flows to carry the reconfiguration information. To update the map of one port of the on-chip interconnect, total 8 bytes are required (4 bytes for data and 4 bytes for address). The term $FPS \times H$ represents the size of data occupied by the network header.

Otherwise, if the average packet size of the traffic is given, we can use equation 2 to represent the relationship between the dynamic of network and the data rate of *SNFR* flows

- S: Average packet size.

$$8 \times p = r \times \left(1 - \frac{H}{S}\right) \tag{2}$$

The term $\left(1 - \frac{H}{S}\right)$ represents the ratio of actual size of data for reconfiguration information over the total size of data.

if we assume a *SNFR* flow encapsulated in standard IP over Ethernet packet ($H = 34$ bytes) at data rate of 100 mbps ($r = 100$ mbps) with random packet size ranging from 64 bytes to 1518 bytes, and with average packet size 791 bytes ($S = 791$ bytes), according to equation 2, it is possible to change the map of port of the on-chip interconnect around 1.5 million times every second ($p \approx 1.5 \times 10^6$). It should be pointed out that the 1.5 million times is the total number of updates of port map on all the associated FPGA devices in the network. For example, the 1.5 million updates can be performed on a single FPGA device every second. Or they can be performed on 1.5 million FPGA devices and once per second per device.

VI. Conclusion

The evolution of data center architecture tends to disaggregate data center resources. This paper proposes a method for the disaggregation of FPGA resources in the data center. The motivation to synchronize the reconfiguration of coherent functions on disaggregated FPGA resources is explained. A network protocol named *SNFR* to synchronize the reconfiguration is proposed in this paper. The associated protocol processor is implemented. The synchronized function reconfiguration process on disaggregated FPGA resources is demonstrated. The reconfiguration is captured by the Xilinx debug core and external traffic analyzer. The implemented protocol processor is able to support a high data rate traffic of up to 9 Gbps Ethernet traffic (8.1 Gbps for regular flows and 0.9 Gbps for *SNFR* flows).

Acknowledgments

This work is supported by the EC H2020 dredbox project with grant agreement No.687632.

References

[1] K. Katrinis, G. Zervas, D. Pnevmatikatos, D. Syrivelis, T. Alexoudi, D. Theodoropoulos, D. Raho, C. Pinto, F. Espina, S. Lopez-Buedo, Q. Chen, M. Nemirovsky, D. Roca, H. Klos, and T. Berends, "On interconnecting and orchestrating components in disaggregated data centers: The dredbox project vision," in *2016 European Conference on Networks and Communications (EuCNC)*, June 2016, pp. 235–239.

[2] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–10.

[3] F. Chen, Y. Shan, Y. Zhang, Y. Wang, H. Franke, X. Chang, and K. Wang, "Enabling fpgas in the cloud," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, ser. CF '14. New York, NY, USA: ACM, 2014, pp. 3:1–3:10. [Online]. Available: http://doi.acm.org/10.1145/2597917.2597929

[4] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling fpgas in hyperscale data centers," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, Aug 2015, pp. 1078–1086.

[5] O. Segal, M. Margala, S. R. Chalamalasetti, and M. Wright, "High level programming framework for fpgas in the data center," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2014, pp. 1–4.

[6] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Disaggregated fpgas: Network performance comparison against bare-metal servers, virtual machines and linux containers," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2016, pp. 9–17.

[7] L. Pezzarossa, M. Schoeberl, and J. Spars, "Reconfiguration in fpga-based multi-core platforms for hard real-time applications," in *2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2016, pp. 1–8.

[8] openstack. [Online]. Available: www.openstack.org

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

[10] O. Knodel, P. Lehmann, and R. G. Spallek, "Rc3e: Reconfigurable accelerators in data centres and their provision by adapted service models," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, June 2016, pp. 19–26.

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

[12] Q. Chen, V. Mishra, and G. Zervas, "Reconfigurable computing for network function virtualization: A protocol independent switch," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Nov 2016, pp. 1–6.