Author:
Chen, Qianqiao

Title:
Exploring hardware support for resource management in the data centre

Exploring hardware support for resource management in the data centre

By

QIANQIAO CHEN

Department of Electric and Electronic Engineering
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Electric and Electronic Engineering.

MAY 2018

Word count: Approximately 22000

Abstract

The management of data centre resources has become important as their size, complexity, and power consumption are increasing. Efficient resource utilisation helps reduce the construction and maintenance costs of data centres. Using that background as a starting point, this thesis explores hardware support that increases the efficiency of resources in data centres.

Either optimising existing data centre architectures or establishing new data centre architectures has a strong requirement on the data centre network. Therefore, this thesis establishes a network interface equipped with the new OptoPHY optical transceiver. The feasibility and quality of an optical transceiver when it is combined with a server board are evaluated by comparing it with a traditional SFP+ transceiver.

The reconfigurable FPGA platforms have a great potential to reduce power consumption and maintenance costs due to their high-volume programmable parallel processing features. Therefore, this thesis tries to explore the management of FPGA resources in a data centre environment.

Based on partial reconfiguration technology, an architecture that would virtualise the FPGA platforms is proposed and implemented. Resources on the FPGA chip are partitioned into several regions. Furthermore, an interconnect system is implemented to share I/Os with all of the regions and enable communication between regions.

When several FPGA platforms are combined for pipelined stream processing, it is difficult to perform function reconfiguration over multiple FPGA platforms that have a short downtime. That is because it is hard to predict the network delay of reconfiguration requests sent from a centralised manager. Therefore, a new network protocol is proposed, and the associated protocol processor is implemented to synchronise the reconfigurations of related FPGA platforms.

Dedication and acknowledgements

I would like to thank my supervisors, Dr. Georgios Zervas and Dr. Jose Nunez-Yanez, who have helped me with their guidance, supervision, and constructive comments. My PhD has been an amazing experience, and I sincerely thank them for the tremendous academic support they have given me.

I would also like to thank my colleagues Vaibhawa Mishra, Hui Yuan, and Adaranijo Peters. I have had plenty of helpful discussions with them at various stages of my studies.

I would like to thank my parents. Their love and affection have been the sources of motivation and encouragement for my studies. I would also like to thank all my friends for supporting and encouraging me with their best wishes.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

[redacted]

Table of Contents

List of Tables

Table **Page**

LIST OF FIGURES

Chapter 1 Introduction

1.1 Motivation

The information technology (IT) industry has experienced exponential growth since its birth in the previous century to become one of the most remarkable industries in today's world. Cloud computing is shaping the way IT solutions are designed and purchased because of its potential to reduce operating costs and management overheads. Many companies are moving their IT infrastructure from public data centres and are starting to build private data centres. Consequently, the requirement on computing resource such as CPU, memories and storage is rapidly growing with the ever-growing workload of data centres.

Due to the diverse and dynamic demand from tenants, the resource utilisation found in data centres has become inefficient. Studies show that from June 2009 to May 2011, the average utilisation of CPUs, memory , and discs on thousands of randomly selected data centre servers was 17.76%, 77.93% and 75.25% respectively [1]. While the covariance was 1.02%, 0.3% and 0.32% respectively. Another study on resource utilisation of the Google Cluster Trace over a period of 29 days reports that the mean CPU utilisation is less than 60% and the mean memory utilization is less than 50% [2]. The allocated CPU resources to a task equals the peak requirement (70% of a server), although the CPU requirement of a task is 40% most of the time. According to these surveys, data center operators tends to allocate resource according to the peak resource usage of a task or a request. And the reason is that current data center architecture cannot support smooth resource reallocation during the runtime of a task.

The inefficient resource utilisation in a data centre results in increasing economic pressure for a firm. It is reported that in 2015, more than 400,000 data centres in China alone were consuming about 1.5% of national electricity supply [3] [4]. In the U.S., consumption in 2014 reached 70 TWh, about 1.8% of total supply [5]. Energy consumption caused by the inefficient resource utilisation also increases the operating costs for data centre operators. Furthermore, it is predicted that densely populated data centres may arouse environmental problems due to their tremendous energy consumption [6]. To optimise resource utilisation, research and technologies to better manage the fast-growing data centre resources are proposed. For example, the resource virtualisation propose methods to share a single physical platforms to multiple users, and the resource disaggregation suggests to combine distributed platforms to build a complete computing system. In order to increase the resource efficiency of data center, this paper focus on developing methods and platforms to deliver high bandwidth flexible network stream processing.

Servers in data centres act as the main containers of computing resources. Cloud applications are deployed on servers in the form of computing resources requests. Operations within each server are organized automatically by local operating systems, while operations between servers require an additional data centre manager. In other words, operations between servers are more complex and have a weaker performance than operations within servers. To avoid operations between servers as much as possible, data centre operators usually abstract the resource management in data centres into a bin pack problem where cloud requests are packed by servers in a best-fit way. In other words, data centre operators are developing additional algorithms and building additional control layers in resource management to overcome the low performance of operations between servers. Difficulties in operations between servers come from hardware restrictions. Networks within a server only need to connect a limited number of endpoints, while networks between servers (data centre network) ideally need to connect countless endpoints. Compared to networks within servers, data centre networks have stronger performance requirements. Therefore, this thesis explores the network interface to increase the performance of data centre networks and build a data centre resource with fewer boundaries caused by servers.

On the other hand, heterogeneous data centers, which contains a set of accelerators for graphic processing and reconfigurable processing, have been suggested. Reconfigurable devices are able to deliver application-specific higher performance functions compared to general purpose processors. At the same time, they are more flexible than other application-specific integrated circuits. These advantages of reconfigurable FPGA resources help reduce power consumption and increase the performance of data centres. However, the reconfiguration feature also provides flexibilities in the control of reconfigurable devices, which introduces more challenges for the management of reconfigurable resources.

The reconfigurable FPGA resources are often treated as a piece of memory that is attached to general purpose processors since existing resource management methods such as virtualisation can be adopted. However, parallel high-volume stream processing is a key feature of FPGA resources. Data movement through the processors will affect this feature as the memory-mapped interfaces between processors and FPGA resources limit the total performance. Therefore, this thesis will try to develop a system that virtualises and manages FPGA resources independently without involving data movement through memory-mapped interfaces.

Apart from data movement, the function reconfiguration on multiple devices also needs to be concerned with the management of reconfigurable FPGA resources. Existing solutions that reconfigure functions on multiple FPGAs follow a deploy-and-run manner. Only when all the associated FPGA devices are properly configured can the newly-deployed function begin processing. As the downtime of every reconfiguration is relatively long, it is hard to perform function reconfiguration at a very high frequency. This is also because most of the time will be consumed by function reconfiguration rather than data processing. Therefore, this thesis will try to come up with a new reconfiguration manner to reduce downtime and to enable reconfiguration at higher frequencies.

1.2 Goals

In summary, the purpose of this thesis is to explore and develop hardware systems that help increase the resource efficiency in data centres. This thesis aims to propose hardware systems that increase the performance of data centre networks, as well as to proposes methods that introduce reconfiguration FPGA resources in data centres. The goals of this thesis are listed as follows:

- Introduce the development of resource management in data centres, including virtualisation and disaggregation of computing resources.
- Develop hardware systems that increase the performance of data centre networks and help break the limits of servers.
- Design systems to efficiently manage stream processing in reconfigurable FPGA resources at a low performance cost.
- Explore methods to control the reconfiguration of multiple FPGA resources in data centres.

1.3 Thesis outline

Chapter 2 reviews the literature on data centre resource management. According to the purpose, this thesis classifies the management method into two types: virtualisation and disaggregation. Virtualisation tries to partition resources on a single physical machine and share the resources with multiple tenants. Meanwhile, disaggregation aims to combine resources on multiple platforms into a complete system. Specifically, the virtualisation and disaggregation of FPGA resources are reviewed. Methods that manage FPGA resources in a data centre environment are introduced.

The burden of the scale-up of data centres mainly falls on the network. Increasing the performance of existing data centres and developing new data centre architectures both require a powerful high bandwidth network. Therefore in chapter 3, a high bandwidth network interface for the server is set up. Equipped with advanced optical transceivers, the network interface is able to deliver a higher bandwidth at lower power costs and a lower footprint. The bit error rate of an optical transceiver is collected and is compared with the that of a traditional SFP+ transceiver. Furthermore, the interface is tested in the environment of an optical data centre network by attaching it to an optical switch.

The virtualisation of reconfigurable FPGA resources is explored in chapter 4. Based on partial reconfiguration technology, resources on an FPGA platform are partitioned and distributed to different users. By establishing interconnect structures , the I/Os such as Ethernet ports are shared by reconfigurable regions and enable communication with off-chip networks. As an example, this thesis demonstrates the use of cases in the virtualized FPGA platform in the area of network function virtualisation to deliver high-performance flexible network functions.

Chapter 5 explores the disaggregation of reconfigurable FPGA resources. A common method to manage and combine FPGA resources is to send a configuration request to each FPGA device from a centralised controller. However, as network latency is not predictable, it takes time to deploy functions resulting in downtime during the runtime reconfiguration. Therefore, in chapter 5, a new network protocol is suggested to spread configuration requests through the data centre. The protocol is able to synchronise the reconfiguration on pipelined stream processing over multiple FPGA resources.

Chapter 6 states the conclusion of the thesis and summarises the work that was done.

1.4 Main contributions

This thesis explores hardware systems that support resource management in data centres. This work also proposes methods that both increase the performance of data centre networks and those that introduce reconfigurable FPGA resources in data centres. The main contributions are summarised as follows:

- Combine an OptoPHY optical transceiver with FPGA devices to establish a high-bandwidth network interface at a low PCB footprint cost.
  - Compare OptoPHY with a traditional SFP+ optical transceiver.

  - Set up the optical switch to emulate the optical data centre network.

- Implement a system to virtualise the FPGA platform. The feature of high-volume stream processing can be kept in the said system.
  - Establish an on-chip interconnect system to switch traffic based on the Ethernet address.
  - Propose a process to enable reconfiguration with a low downtime.
- Design and implement a method and an associated network protocol to synchronise the reconfiguration over multiple FPGAs.
  - A network protocol to synchronise reconfiguration is designed.
  - The associated protocol processor is implemented.
  - The high-frequency reconfiguration over multiple FPGA devices is demonstrated.

1.5 Publications

All the work is done with the help of my colleagues Vaibhawa Mishra, Hui Yuan, Adaranijo Peters, Peter De Dobbelaere, Michael Enrico, and Nick Parsons, as well as my supervisors, Jose Nunez-Yanez and Georgios Zervas. The contribution is supported by a number of academic papers:

Q. Chen, V. Mishra, N. Parsons, and G. Zervas, "Hardware programmable network function service chain on optical rack-scale data centers," in 2017 Optical Fiber Communications Conference and Exhibition (OFC), 2017.

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), 2016.

Q. Chen, V. Mishra, and G. Zervas, "Synchronizing reconfiguration of coherent functions on disaggregated FPGA resources," in 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2017.

Q. Chen, V. Mishra, G. Zervas,"SiP-enabled FPGA Network Interface for Programmable Access to Disaggregated Data Centre Resources", OSA Asia Communications and Photonics Conference, Nov 2017

Q. Chen, V. Mishra, J. Nunez-Yanez, and G. Zervas, "Reconfigurable Network Stream Processing on Virtualized FPGA Resources," International Journal of Reconfigurable Computing, vol. 2018, Article ID 8785903, 11 pages, 2018. doi:10.1155/2018/8785903

V. Mishra, Q. Chen, and G. Zervas, "REoN: A protocol for reliable software-defined FPGA partial reconfiguration over network," in 2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2016, pp. 1–7.

G. Zervas, V. Mishra, Q. Chen, "Network, Compute and Storage Function Programmability and Virtualization: An FPGA-based Disaggregated System", OSA Asia Communications and Photonics Conference, Nov 2016

G. Zervas et al., "Disaggregated compute, memory and network systems: A new era for optical data centre architectures," in 2017 Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1–3.

G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, "Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [invited]," IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 2, pp. A270–A285, Feb 2018.

K. Katrinis et al., "On interconnecting and orchestrating components in disaggregated data centers: The dReDBox project vision," in 2016 European Conference on Networks and Communications (EuCNC), 2016, pp. 235–239.

K. Katrinis et al., "Rack-scale disaggregated cloud data centers: The dReDBox project vision," in 2016 Design, Automation Test in Europe Conference Exhibition (DATE), 2016, pp. 690–695.

Chapter 2

Literature review

2.1 Introduction

Virtualisation is widely adopted to manage computing resources. It partitions resources on a single physical machine and shares them with multiple tenants by providing each tenant with an independent interface. Recent research also shows interest in the disaggregation of data centre resources. Although there is no final conclusion on the definition of resource disaggregation, this thesis defines disaggregation as technologies that combine resources on multiple platforms into a single complete computing system. The research on the management of data centre resources is reviewed in this chapter. Existing technologies on resource virtualisation are discussed. The new disaggregation concept is then introduced.

The resource virtualisation originates from the instruction-by-instruction simulation of a computing system on a different system. At the very beginning (in the middle of 1960s), virtualisation was often used by programmers who were working on a more general-purpose computer but were at the same time developing software for a new special-purpose computer which was under construction. They likely began by writing a simulator for the new computer on other available machines. Programmers were soon aware that it is possible to run a number of copies of the simulator at the same time and build a time-sharing system that supports multiple users [7].

Since then, virtualisation has evolved into the most common technology to manage computing resources. It is able to decouple the software from the physical hardware platform by delivering a diverse virtual environment to its client applications. Moreover, it can also divide the physical machine by (a) time-sharing the CPU resources and (b) partitioning the storage resources [8].

Being another technology that manages data centre resources, resource disaggregation recently attracted the attention of researchers. It originates from the concept of virtual storage [9]. Computer architecture designers have found that it is difficult to integrate large amounts of storage inside the chip. Hence, they started to use external storage and designed mechanisms to automatically swap the data in the external storage with the data in the internal storage. This approach enables the hierarchy storage systems. Based on observations of imbalance density growth of different on-board resources (e.g. CPU, memory, discs), there are reports that the relative low-density on-board memory limits the overall performance of a computer system [10].

On the other hand, data centres (previously known as multicomputers) whose performances are scalable with the number of computer resources have evolved out of the research [11]. Specifically, heterogeneous servers equipped with additional specific resources were widely adopted in data centres. Meanwhile, mechanisms which enable access to remote memory are proposed [12]. Heterogeneous servers in data centres have become the solution to the imbalance in the density of different on-board resources. Access to the remote memory was initially organised by the communication between local and remote processors. However, to reduce access latency and bypass the remote processors, remote direct memory access (RDMA) is proposed. The heterogeneous servers and RDMA have evolved into today's data centre resource disaggregation.

The disaggregation enables the independent scale-up of different types of resources by combining computing resources found on physically distributed platforms into a computing system. With resource disaggregation, data centre operators can upgrade different resources independently. Resource disaggregation also enables multiplex resources beyond the edge of the board at the scale of a data centre, which can potentially increase the efficiency of resource utilisation.

Both the virtualisation and disaggregation aim to project an illusion of machine interfaces.

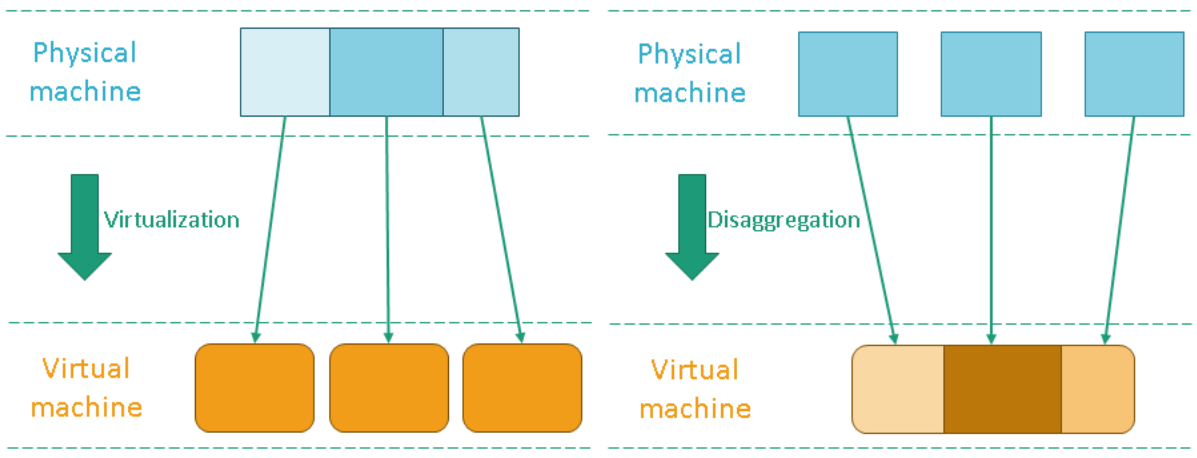

FIGURE 2.1. The resource virtualisation and disaggregation of data centres.

However, different methods are adopted by the two concepts to achieve their goal. Virtualisation tries to isolate computing resources on a single physical machine to form multiple machine interfaces. Ideally, when using one of the interfaces, the user cannot feel the existence of other interfaces. On the other hand, disaggregation attempts to combine computing resources on multiple physical machines to build an interface out of a single machine. The user of the interface should not feel the location or distribution of the physical resources. In summary, as shown in Figure 2.1, virtualisation talks about how to transform a single physical machine into an illusion of multiple single machines, whereas disaggregation discusses how to combine multiple physical machines to deliver the illusion of a single machine.

While the virtualisation and disaggregation are independent concepts, they can create a powerful solution to the resource inefficiency of data centres when combined together as shown in Figure 2.3. The virtualisation technology is able to divide computing resource into fine-grained virtual infrastructure. With disaggregation, it becomes possible to interconnect these infrastructure into on demand systems. Consequently, the combination of virtualisation and disaggregation helps to get every piece of data centre resource involved in the cloud computing.

This chapter reviews the associated technology for the resource virtualisation and disaggregation. The rest of this chapter is arranged as follows: in section 2.2, the virtualisation technology is reviewed. Section 2.3 introduces the disaggregation of data centre resources. Especially, the

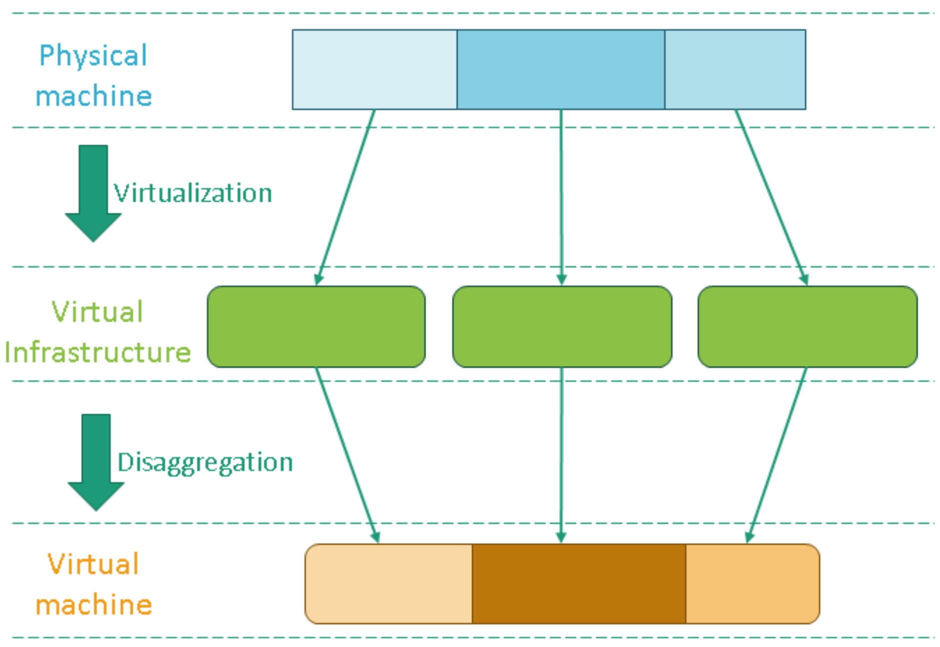

FIGURE 2.2. The combined utilization of virtualisation and disaggregation.

virtualisation and disaggregation of FPGA resources is reviewed in section 2.4. And section 2.5 summarises this chapter.

2.2 Resource virtualization

Virtualisation is the most important technology in resource management of data centres. It divides a physical machine into multiple independent virtual machines by introducing a virtual machine monitor or hypervisor on top of the hardware platforms. Each virtual machine ideally executes its own operating system and applications without affecting others. Resource management based on virtualisation usually includes three steps: prediction, allocation, and migration. These three steps will be introduced in this section.

As the workloads of data centres fluctuate over time, the allocation of resources appropriate for a particular time slot may become inefficient in another time. As the following resource allocation algorithm and virtual machine migration may take a long time, it is hard to deploy applications immediately. Therefore, the prediction of a data centre's workload is essential in the management of its resources.

2.2.1 Prediction

Jheng et. al. [13] find that the virtual machine (VM) workload has a strong periodicity, so they predict the workload by tracing the historical workload of each VM in the data centre. Some researchers notice that not all VMs observe the same workload pattern, thus they try to classify the VMs. Khan et. al. decompose applications into multiple components and each component is served by a set of VMs [14]. Algorithms are applied separately to VM sets that serve different components to predict the workload more accurately. Similarly, Chen classifies the VMs based on the tenant and finds that workloads from the same tenant are similar [15]. However, Bobroff states that not all the VMs' workloads are predictable, i.e., some VMs' workloads either lack periodicity or experience random changes over time [16]. Some scholars have put forward more advanced models. For example, Kashifuddin adopts the chaotic theory in the prediction of resource utilization [17], while Zhen's research utilises the Exponentially Weighted Moving Average (EWMA) model [18].

Although some scholars have suggested various models, predicting the workload of short-running tasks is still complicated and takes time. In contrast, long-running tasks show a periodic feature and are more predictable. Consequently, most workload predictions are only applicable in a relatively static scenario. For a dynamic data centre, it is hard for the prediction algorithm to come up with an accurate result on time.

Apart from monitoring resource utilisation, methods based on the trace of incoming requests are also suggested. For example, Vercauteren et. al. [19] trace the number of requests in web servers that randomly fluctuate over the short-term but exhibit periodic patterns in a diurnal cycle. Fang et. al. [20] have applied the ARIMA model to predict the short-term workload based on observing the number of requests. Unlike the resource utilisation of VMs, the number of incoming requests are easier to predict. However, it is difficult to map the incoming requests to the actual workload. These requests on the target hardware platforms need to be tested in advance, which needs additional effort. The interference between collocated requests has also not been considered, which may complicate the prediction of workloads.

In summary, short-term workloads are challenging to predict. If the data centre is running applications with very dynamic hardware resource requests, the prediction may become inaccurate. As the input of the allocation, large prediction errors may mislead the following steps of resource management.

2.2.2 Allocation

After predicting the workloads, algorithms are required to allocate resources to the incoming applications. Allocation algorithms are responsible for placing VMs into servers with efficient resource utilisation (minimum active physical machines and maximum resource utilisation on each server).

The placement of VMs is abstracted into a bin-pack problem in [21] [22] as follows:

- Givens: (1) An N number of VMs and the resource demand of each VM. (2) The resource capacity of each server.
- Obtain: The location of each VM.
- Objective: Reduce the number of active servers.
- Constraints: (1) Each VM is placed on only one server. (2) the utilization does not exceed the capacity of all servers

Many algorithms exist to solve the bin-packing problem such as Best Fit, First Fit, etc. But the results of the model heavily depend on the preciseness of the prediction, as any random events are not taken into account. To solve this problem, Jin et al. [23] classify the resource demands by VMs into static and dynamic parts. The static demands are treated as a constant value, while the dynamic demands are formulated as a random number of normal distributions. Moreover, an additional constraint is added to the aforementioned bin-packing problem: for each VM resource demand, the under-provision probability should be less than a threshold.

Considering the placement of VMs as a bin-packing problem results in the relocation of all the VMs. Consequently, the migration of VMs takes a very long time because most VMs need to be migrated in every time slot. Therefore, in order to reduce the number of instances of migration, some research suggests a method that can be applied in real time.

Most real-time algorithms try to rearrange VMs from high-load servers to low load servers one-by-one. The low-load servers are ordered based on a specific standard. The movement of VMs is performed starting from the lowest load server on the list, and it continues until the list of high-load servers is exhausted [24]. This method, however, needs to traverse all type of resources on every server to find a suitable low-load server, which takes a long time. Therefore, the load of the server is abstracted into a load indicator function to enable these algorithms in real-time allocation. However, it is hard to come up with an accurate function to indicate the thorough load of a server.

In summary, because of the heterogeneous servers and the resource demands of VMs, the allocation algorithms often need to compare between huge tables with multiple columns. As it often takes a long time to finish the comparison, having accurate allocation algorithms in real time becomes difficult. Most allocation algorithms assume that it is impossible to divide a VM into multiple parts and locate them on different servers. That is the main reason why they try to compare the VM resource demand tables and server resource capacity tables to find a suitable placement of VMs. As a result, investigating how to enable distributed VM leads to the concept of resource disaggregation. The technology of resource disaggregation makes the distribution of VM possible and helps create faster VM allocations.

2.2.3 Migration

After allocation algorithms, the data centre should be able to evolve into a better optimised state. Instead of fixing VMs on the server where they were created, VM migration enables the movement of VMs from one server to another based on the allocation result.

The general idea of VM migration is to copy memory from the source server to the destination server and keep the function of the VM running as smoothly as possible. For example, while the memory is moving, the performance of the VMs should not be affected and there should be less downtime. It needs to be pointed out that, if the VM keeps functioning during the migration to reduce migration downtime, there will be differences between the source memory and the copied memory, as the source memory might be updated when the copied memory data is moving. These differences between source and destination should be handled while VM migration is going on.

During the migration process, the original memory data is copied first to the destination server. Then the newly-dirtied data is iteratively transferred to the destination. When the iteration reaches a specific threshold, the source VM will be stopped. The remaining dirtied data will then be packed and transferred to the destination server together. Finally, the VM in the destination server will be activated.

Many technologies are adopted to accelerate the VM migration. For example, to reduce the network traffic, some research suggests compressing data in the memory before the transmission, as the memory pages have strong regularity [25]. These methods need to make a trade-off between computing performance and migration performance, as compression needs additional computing. High-speed connection technologies can also accelerate the VM migration by providing a dedicated bandwidth and specific remote direct memory access functions.

VM migration and technologies associated with it explore the handling of memories on a remote server. These technologies not only enhance the resource virtualisation by accelerating the migration process but also enlighten the concept of resource disaggregation. Instead of accessing remote memory for a short time during the iteration of transferring dirty memory data in the migration, it is also possible to keep accessing remote memory during a VM's lifespan. However, this approach has stronger requirements on the performance of the network.

2.2.4 Summary

Virtualisation technology is reviewed in this section. Virtualisation can be divided into three steps: workload prediction, resource allocation, and VM migration. It is difficult to predict the short-term workload of data centres with dynamic applications. As it is unable to distribute a single VM on multiple servers, the allocation algorithms need to handle huge tables that record different types of available server resources to find a suitable place for a VM. The VM migration technology explores methods that handle remote memories which become key enablers of resource disaggregation.

2.3 Resource disaggregation

To better utilise data centre resources, the concept of resource disaggregation is proposed. Instead of tightly binding all types of resources (processors, memories, etc.) on a single board, resource disaggregation tries to connect computing resources located on separated boards into a system. Ideally, as the location of the resources (local or remote) does not affect the performance of the resulting system, it would be able to equip a board with a unique type of resource. This feature can potentially contribute to the operation and upgrading of data centres, as different types of resource are managed and scaled independently. However, as it is impossible to avoid the overhead introduced by the interconnect system, performance penalty of resource disaggregation becomes a fact of life. As a result, combining distributed resources into a system at a low performance cost becomes the key question in resource disaggregation.

2.3.1 Software support

To demonstrate the concept of resource disaggregation, a software solution of the resource disaggregation is suggested as an initial step. Regardless of performance, with the software solution, executing an application on disaggregated resources is possible.

Software systems for memory disaggregation are first implemented. Lim et al. [10] implement a software prototype with a processor and a remote memory blade linked by PCIe. Instead of a point-to-point system, a PCIe switch is adopted in [10] to extend the prototype into a many-to-many system. Apart from PCIe, the FaRM [26] presents a scalable system where servers are interconnected by RDMA over Converged Ethernet (RoCE) [27]. Then, a complete OS level solution [28] is proposed as a further step to resource disaggregation.

The software solution aims to provide an efficient system to manage local and remote memory. It enables the cloud application developer to design and run their program on a resource-disaggregated environment. However, due to the overhead from the interconnect , applications that run in the resource-disaggregated environment suffer poor performance, while software solutions contribute little to performance improvement [29].

2.3.2 Hardware support

As the main overhead cost comes from the interconnected system, a data centre network becomes the key element to reduce the performance penalty from resource disaggregation. As the traffic previously existing within a server is moved to the external network, bandwidth and latency become a key requirement in resource disaggregation [30]. Since various additional protocols such as RDMA are often introduced to support disaggregation, the flexibility of a network is also another important element.

Optical networks have become a promising solution to address the challenges in resource disaggregation due to their high network capacity and low energy consumption [31]. Advanced photonic technologies also enable switching at the optical domain, which eliminates the latency and energy introduced by electrical switches [32]. However, the advanced photonic technologies often need specialised optical interfaces. These optical interfaces may introduce challenges in the board design of the servers as they often require additional power and footprint. Therefore, to be compatible with the advanced optical network, Chapter 2 proposes new network interfaces that consider bandwidth density and footprint size.

Due to the non-availability of an optical buffer solution, the packet-level processing of a data centre network is based on electronic devices. Recent research on packet-level network processing focuses on increasing the flexibility of the network to make it easier to manage, upgrade, and evolve. As a consequence, a software-defined network (SDN) is proposed [33].

The SDN tries to decouple the control and data plane of the network. The infrastructure of SDN is shown in figure, there are two elements, the controllers and the data plane devices. The data plane devices is a hardware or software element specialized for traffic forwarding, while the controller is a software element executed by a commodity hardware platform. A SDN device is based on a pipelined of flow forwarding tables with the following entries: a) a matching rule; b) actions to be executed after packet matching and c) counters that record statistics of packet matching. In the SDN devices, a path through a sequence of flow tables defines how packets should be processed. The possible actions performed on packets includes: 1) forward the packet to output ports; 2) forward the packet to the controller; 3) drop the packet and 4) forward the packet to a specific flow table.

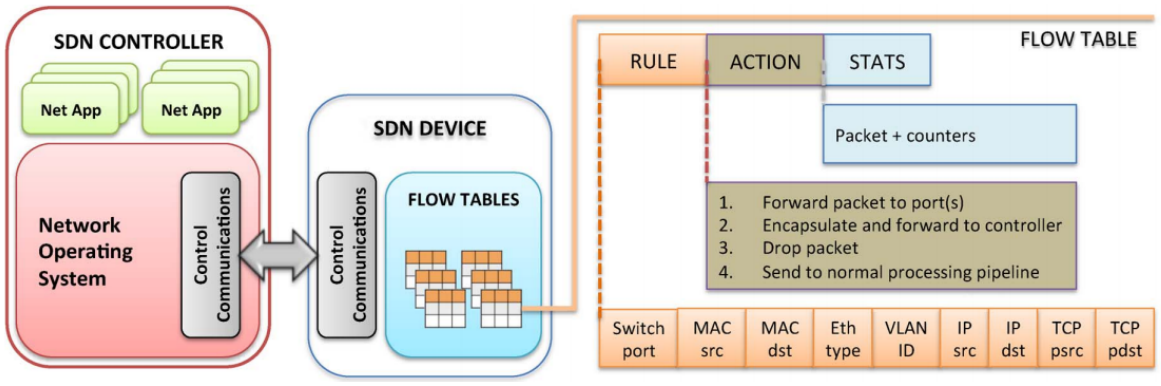

FIGURE 2.3. The overall concept of software-defined network. [33]

The control and management on the distributed data plane of network is centralized in the SDN. Although the SDN suggests a software control plane only, the software data plane is also implemented as a complement to the SDN concept. In data centres, software network switches are often adopted to forward data between virtual machines in the same sever.

Theses software applications on network control and data plane implicates the concept of network function virtualization (NFV). In the NFV, network functions are treated as software applications that run in the operating system. So virtualization technology can be applied to virtualize the network resource. The NFV inherited many advantages from SDN. For example, network services can be deployed easily in NFV. The NFV also has its own advantages such as sharing physical network resources to multiple tenants of data centres.

Although running software network applications increases the flexibility of a network, general purpose processors may limit the performance of these network functions especially on the data plane. An FPGA, as a reconfigurable device, becomes an outstanding hardware platform to keep both the flexibility and performance. In Chapter 4, a virtualise system is implemented on the FPGA platform to deliver a flexible high-performance network function.

2.3.3 The dRedBox

The dRedBox research project is one of the ongoing research projects which investigates resource disaggregation [34]. It aims to deliver a customisable data centre architecture by introducing advanced technologies in optical networks to eliminate the effect from the boundaries of the server board.

The architecture of the resource disaggregation architecture in dRedBox is shown in figure 2.4 [35]. The disaggregated racks (dRack) contain many disaggregated boxes (dBoxes). Each dBox includes a set of pluggable disaggregated bricks (dBrick) equipped with a specific computing resource. Each dBrick can support unique types of resources such as general purpose processors, memories, or accelerators of a specific application. All dBricks are able to have a direct connection to other dBricks through electronic or optical switches. The optical circuit switches mainly connect dBricks located on different dBoxes, while the electronic crossbar switch is responsible for the connection between dBricks on the same dBox.

Apart from the main computing resource, as the glue logic between dBricks and dBoxes, each dBrick is also equipped with a reconfigurable system to deliver on-demand high-performance network functions such as network interfacing and network switching. For example, the reconfigurable system needs to provide RDMA protocols when it is interfacing processors with memories and IP protocols when interfacing processors with end users. In Chapter 3, a reconfigurable system to deliver on-demand network functions is implemented and will be adopted as the glue logic between the dBrick and dBoxes.

Each dBrick is also equipped with an onboard optical transceiver. To minimise the footprint size and power consumption while increasing bandwidth, an optical transceiver based on advanced silicon photonic technology is tested in Chapter 2. The adopted optical transceivers and the optical switches enable a scalable, transparent dBrick-to-dBrick interconnect system with ultra-low latency and high bandwidth.

2.3.4 Summary

The data centre resource disaggregation is reviewed in this section. Software solutions are first proposed to enable the concept of resource disaggregation. They mainly focus on the management of the data in remote memory. Most software solutions have a strong requirement on the network. Therefore, a high-performance data centre network is established by introducing advanced interconnect technologies. To increase bandwidth, the optical network is introduced. The SDN

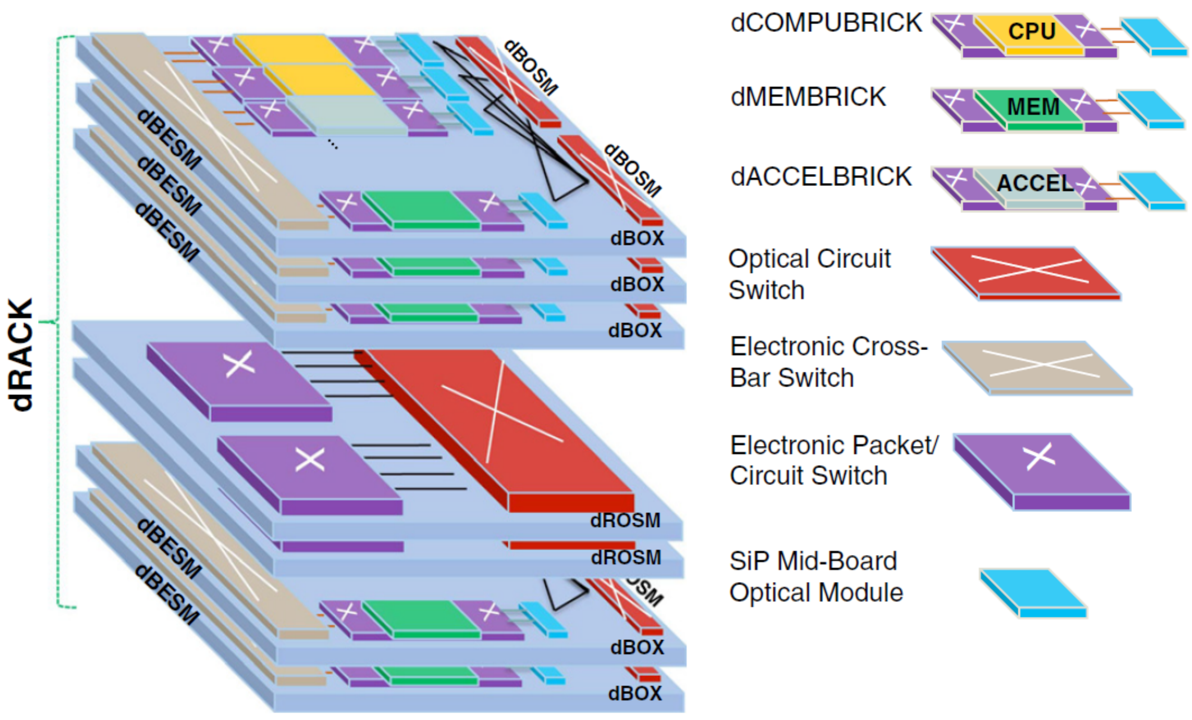

FIGURE 2.4. The structure of the dRedBox data centre resource disaggregation system.[35]

and NFV are also introduced to increase the flexibility of the network.

2.4 Management of FPGA resources

The FPGA devices have been adopted in data centres as reconfigurable resources. Due to their extreme customisation feature, FPGA devices are widely adopted to support high-performance applications that need non-recurring hardware accelerators. Two types of usage of FPGA resources can be observed in the data centre. Firstly, the FPGA performs as a kind of equipment to enable data centre such as network switches or accelerators of specific processes. The second is as a kind of general computing resource which should be delivered to end users directly.

Similar to the memory, the management of FPGA resources is also based on virtualisation and disaggregation. FPGA resources should be isolated and shared properly in virtualisation, while it should be able to combine distributed FPGA resources to compose an on-demand system in disaggregation. Unlike the memory, however, functions deployed on the FPGA resources are sensitive to their physical location. In other words, inappropriate places and routes may result in failed timing in the FPGA while the location of the program in the memory does not make any difference.

Therefore, additional architecture is required to reduce the effect of the location when managing FPGA resources. In this section, architectures or systems of virtualisation and disaggregate FPGA resources are reviewed.

2.4.1 Virtualisation of FPGA resources

FPGA resources should be isolated and shared by multiple tenants in virtualisation. Partial reconfiguration becomes a key enabler of virtualisation of FPGA resources, as it is able to configure a part of the FPGA while it keeps the rest working [36].

The initial effort for partial reconfiguration is reported in [37]. It tries to modify the bit width of multipliers. Because there are overlap in the logic used by several version of multipliers, researches tries to propose methods to only configure the non-overlapping part of the design to reduce the reconfiguration time. Although tools are developed to identify the overlap parts of the circuit and set the reconfiguration parts automatically [38], companies finally develop tools to make FPGA designers to predefine reconfiguration circuits manually. Also the interface between static and dynamic regions should be predefined and is not reconfigurable at run time because it is made through fixed-location LUT.

Apart from hardware implementation, swapping circuits at run time requires software management. This software needs to consider the size of the reconfigurable region, the performance improvement and the overhead to perform the reconfiguration [39]. The sequence of reconfiguration can also be optimized to reduce the overall application latency. The operating system level design to support for placing portions of an application in FPGA then appeared [40]. The FPGA is treated as a co-processor and its reconfiguration are scheduled as software threads of execution [41].

A number of cloud applications has shown benefits of using partial reconfiguration. For example, [42] shows using partial reconfiguration to update the sorting algorithm for large data sets can accelerate the overall data sorting. The partial reconfiguration also helps to develop network applications for protocol and packet filtering modules in network virtualization [43]. It is also used to accelerate SQL database query operations in [44] and FPGA can be reconfigured in milliseconds to enhance performance.

With partial reconfiguration technology, it would be able to partition FPGA resources into many regions and change the function of each region based on requests from users. However, there are two limitations in the partial reconfiguration technology that need to be considered in the virtualisation architecture. First, instead of allocating in runtime, reconfigurable regions should be pre-set in the design's early stages. Secondly, whatever functions are plugged in, the interface (the number and direction of pins) of reconfigurable regions must not change.

The basic architecture to virtualise the FPGA resources is suggested in [45] as shown in Figure 2.5. The run-time module controller acts as the orchestrator of the complete system. The bus is connected with the controller through the arbiter. The arbiter is responsible for controlling the data flow. Functional modules can share data with each other through the bus. A standard interface module named Bus Com is proposed to ensure that every partial reconfigurable module has a uniform interface towards the static region. The partial reconfiguration is performed in the internal configuration access port (ICAP) on the lower-right side of Figure 2.5. The compressed partial bit stream is loaded from the flash memory and pushed to the ICAP after decompression.

By using partial reconfiguration, FPGA resources are partitioned into several regions, and each of the regions can perform different functions independently without affecting another. In addition, a bus structure is applied to enable data transfer between reconfigurable and static regions. The bus structure defines a uniform interface for the reconfigurable regions. Additionally, it also shares resources in the static region such as controllers and I/Os to all reconfigurable regions.

This architecture was originally adopted in an automotive system to deliver functions such as cabin control and rear-view mirror control. The system includes four reconfigurable regions, and the bus is designed for acknowledgement instead of data movement. As it suggests a dynamic architecture that can deliver on-demand FPGA functions, this architecture has been utilised in many fields which need a reconfigurable high-performance system. The basic architecture has been modified as well according to the feature of different applications. For example, this

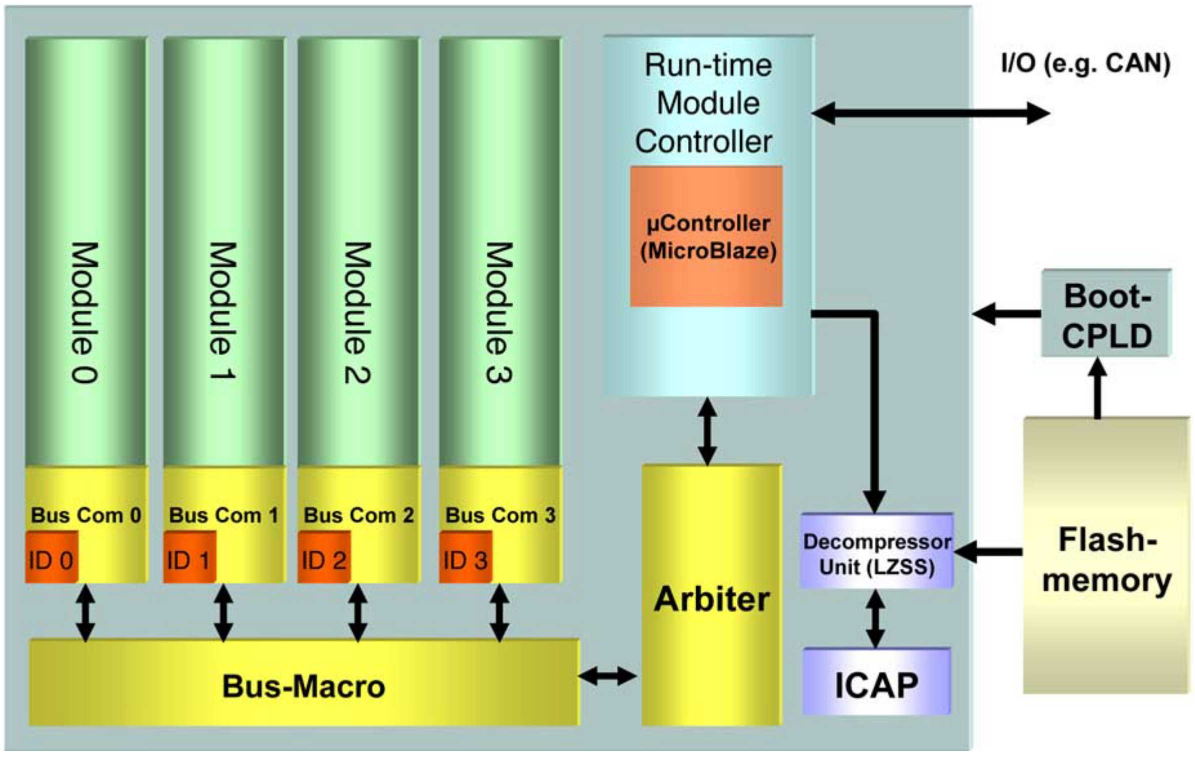

FIGURE 2.5. The basic architecture of the system suggested in [45].

architecture is adopted for video processing in [46]. Two processors are included in the system. One of them is for control of the reconfigurable system, while another one is for the software support of video processing. A memory-mapped bus is also used for data movements between plugged-in modules.

Apart from video processing, the architecture in Figure 2.5 is also used in the processing of network traffic. Because of its stream processing feature, this architecture is able to deliver a high bandwidth and dynamic data plane in recent flexible network architecture technologies [47]. For example, [47] attaches the network interface to the interconnect system and implements partially reconfigurable hardware routers. A bus that supports a 1-Gbps throughput is implemented. Software routers are also deployed in the processor to run as a complementary system during the partial reconfiguration.

Much research has been done to enhance the architecture in Figure 2.5 while applying it in different domains. A network-on-chip system is deployed in [48]. As the timing requirement between reconfigurable regions is released, the network-on-chip helps support more reconfigurable regions and finer-grained partition on the FPGA resources. [49] also suggests allocation algorithms that properly place the reconfigurable regions. The algorithm is able to automatically determine a suitable size and location for each region, and timing constraints in the interconnect system are also considered. [50] and [51] also propose algorithms in the control system that schedule the reconfigurable function in real-time. As these schedule algorithms avoid manually downloading partial bitstreams, they further virtualise the FPGA resources in a time-sharing way.

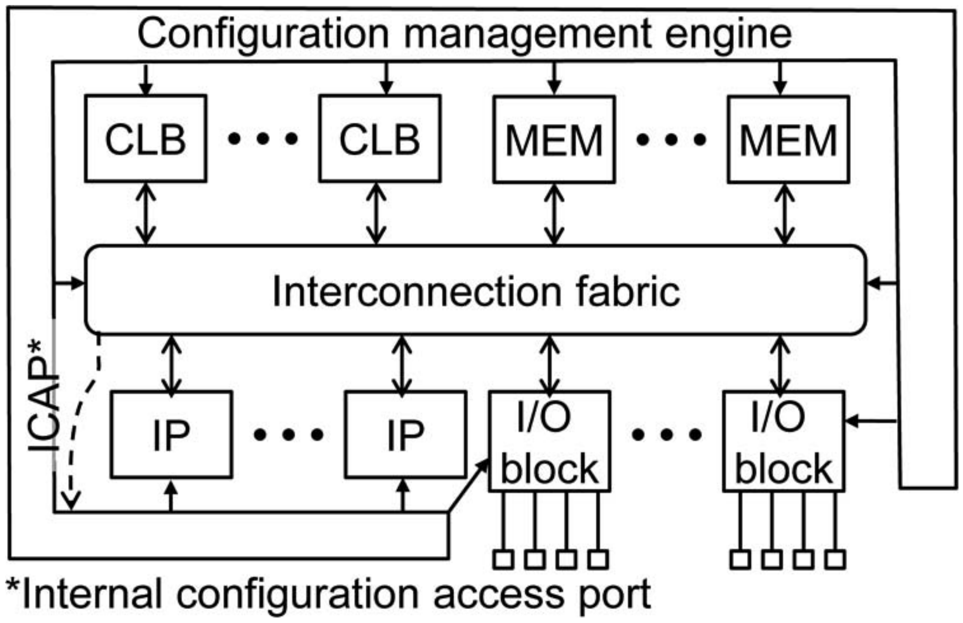
Configuration management engine
CLB
CLB
MEM
MEM
Interconnection fabric
ICAP*
IP
IP
I/O block
I/O block
*Internal configuration access port

FIGURE 2.6. The design principle of the virtualisation system of FPGA resources summarised in [52].

Although the basic architecture becomes complicated after contributions from many functional domains, [52] has summarised the generic design principle of the reconfigurable architecture. As shown in Figure 2.6, three elements should be included:

- Functional blocks

  These functional blocks are able to partition the FPGA resource according to the deployed functions. These functions and their associated FPGA resources should be shared with users after virtualisation.

- Configurable interconnection

  The interconnect system shares data between functional blocks and the external world. It shares routing resources to multiple functional block connections. Shared data is registered and arbitrated in the interconnect system to ease the timing requirement. Therefore, it can reduce the location effects of the functional blocks.

- Configuration Management Engine

  This engine orchestrates the complete system. It updates the port-map of the interconnection and triggers the reconfiguration of functional blocks. It also interfaces the entire FPGA system to the software part.

FPGA resources can be virtualised by following these elements, as they can be managed in a way similar to virtual memory [53]. Based on these principles, a system that virtualises FPGA resources is established and introduced in Chapter 4. In addition, network stream processing functions can be deployed on the virtualised FPGA platform to demonstrate advanced network technologies such as network function virtualisation.

2.4.2 Disaggregation of FPGA resources

When cloud applications are disaggregated and deployed on distributed resources, a virtual platform that combines multiple resources is often delivered. This section reviews research and technologies that combine distributed FPGA resources into a single virtual joint platform.

In [54] and [55], an architecture that attempts to integrate distributed FPGA resources is proposed to support the Bing web search engine. As shown in Figure 2.7, there are two planes in the architecture: a CPU server plane and a hardware acceleration plane. Links between the two planes are enabled by both a Gen3 PCIe interface and a 40-Gbps Ethernet connection. Internal Ethernet links also exist inside each plane. Cloud applications can be distributed on both planes (web search ranking in Figure 2.7). They can also be found on a single plane but with multiple boards (others in the Figure 2.7). This section classifies research in combining FPGA resources based on two types of systems: a)systems that combine FPGA resources through processors. b)systems that combine FPGA resources directly through networks.

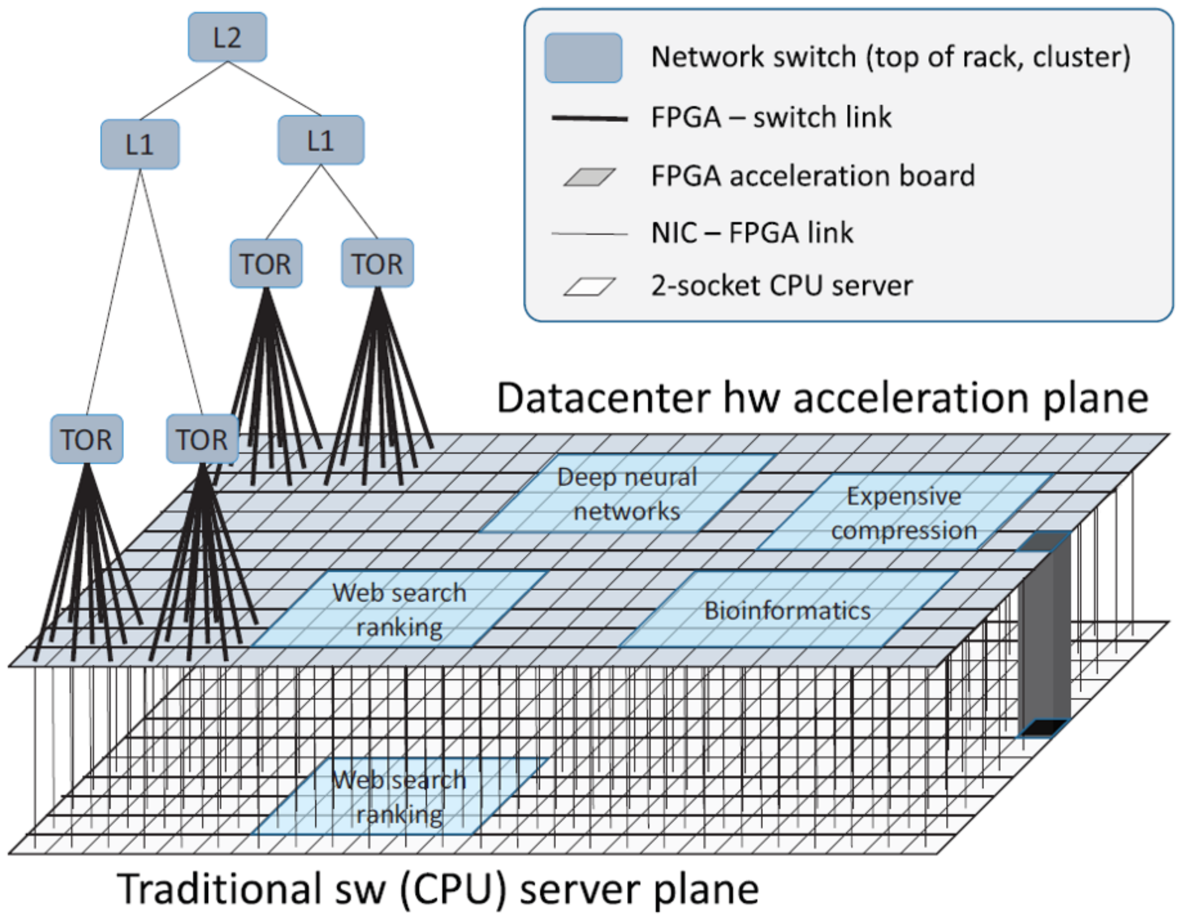

FIGURE 2.7. The disaggregation system for FPGA devices suggested in [55].

In systems that combine FPGA resources through processors, FPGA resources act as complementary accelerators that are attached to general purpose processors. The hardware support of this kind of system is based on the system-on-chip technology or PCIe interfaces. The Xilinx SoC adopts memory-mapped AXI interface for communication between a processing system and programmable logic. The PCIe interfaces are also memory-mapped systems. Therefore, from the perspective of a processing unit, programmable logic can be accessed as a piece of memory.

Consequently, the software support offered by the resource disaggregation through processors tries to apply traditional methods to efficiently manage memories. In [56], the FPGA resources are integrated with the Openstack. The system is mainly designed for stream processing. The data communication between processors and each reconfigurable regions is enabled by a standard memory-mapped interface. A basic library to access control registers or to move memory data in the reconfigurable regions is developed. The Openstack is also adopted in the IBM FPGA cloud system [57]. In addition, IBM developed a hardware scheduler system to reconfigure the FPGA inside the Openstack scheduler. Other than Openstack, a management system designed specifically for FPGA resources called RC3E is proposed in [58]. Functions in the centralised hypervisor system such as job scheduler, compute nodes manager, and user interfaces are developed. The cloud-wide FPGA design that includes HLS is also introduced in RC3E [59].

As the FPGA is considered as an accelerator beside general purpose processors, FPGA resources are accessed in a memory-mapped way in the disaggregation system that combines resource through processors. Mechanisms and architecture in standard data centre resource management solutions can still be utilised by changing the interface from DDR or SATA to PCIe or AXI. However, high-bandwidth heterogeneous parallel processing is one of the features of FPGA resources that stands out the most. In the system combine resources through processors, data moving between FPGA resources have to go through memory-mapped interfaces (between processor and FPGA), which extremely limits the performance of the final combined FPGA systems. Therefore, disaggregation solutions that combine FPGA resources directly through networks are also required.

In systems that combine FPGA resources directly through networks, similar to the disaggregation of memory, the optical network is also adopted. For example, in the Bing search engine, a 40-Gbps QSFP interface is established to enable direct communications with other FPGAs [60]. A lightweight transport layer network processor is also developed. It reduces the latency by eliminating the packet reorder features in a standard TCP/IP network. Another attempt comes from IBM [61]. Standard layer 2 and layer 3 protocols are implemented. Furthermore, memory that is attached to the FPGA is also partitioned and allocated to each reconfigurable region by introducing a hardware memory manager [62]. In addition to the development of a fundamental structure on a single FPGA, [63] tries to establish a global management system that combines multiple FPGA devices by introducing the service chains concept from network function virtualisation. It aims to deploy pipelined functions for stream processing (instead of memory-mapped processing). Technologies in programmable networks such as OpenFlow are utilised to guide data streams to on-demand FPGAs in a specific order.

The FPGA, as a flexible resource, needs runtime configuration. The configuration (update control registers or download bitstream) on multiple FPGA devices is performed through vertical systems in most research. For example, in the BORPH [64] system, the configuration of a control register is triggered by a central controller and through an on-board bus. The BORPH is designed for configuration on multiple FPGA devices but within the board [65]. The network-wide configuration is often integrated with existing data centre management systems that are triggered by a central orchestrator as reviewed in the vertical FPGA resource disaggregation system. During the configuration, as the FPGA blocks that will be updated cannot function, the data flow should be buffered. As a result, the process on data flow between FPGA resources involves waiting - from sending a configuration request to receiving a configuration success response which can last for at least several microseconds. To reduce the configuration time on multiple FPGA resources, this thesis introduces a new configuration method in Chapter 5. The configuration is triggered by network packets through a horizontal resource disaggregation system instead of a vertical system.

2.5 Summary

This chapter has reviewed the research and technologies involving managing data centre resources. Data centre resource management can be classified into two categories (virtualisation and disaggregation) based on the purpose. Virtualisation aims at properly partitioning a single physical platform into multiple virtual environments, while disaggregation attempts to combine distributed resources into a complete system.

The development from virtualisation to disaggregation is then reviewed. In the virtualisation of data centre resources, there are three steps: prediction, allocation, and migration. It is difficult to predict short-term resource utilisation in most algorithms. This inaccuracy may affect the result of the following steps, especially when the resource requirements of cloud applications are highly dynamic. In the allocation, most algorithms assume that only local memory is accessible. Therefore, they try to find out an optimised virtual machine placement that balances the utilisation of all types of computing resources (CPU clock cycles or memory) on each server. After the initial allocation, the process of migration is performed in the runtime of applications. During the migration process, researchers try to synchronise the memory of the source and destination servers for a certain period. This process hints at the key technology of memory disaggregation (accessing the remote memory). To support memory disaggregation, software solutions are first suggested. However, researchers find that a high-bandwidth low-latency connection is required to make the remote storage act like a piece of memory. Therefore, an optical networks is suggested as it is able to provide direct point-to-point low-latency connections. The bandwidth of an optical network depends on the integration of optical transceivers, where the footprint and power consumption of the optical transceivers should be taken into consideration. Therefore, Chapter 3 establishes a network interface where high-bandwidth optical transceivers are integrated with small footprint and power consumption.

The virtualisation and disaggregation of FPGA resources were introduced in Section 2.4. The virtualisation of FPGA resources can be based on the architecture that is shown in Figure 2.5. It can be summarised as a set of functional blocks that are linked by a configurable interconnect system. Based on this design principle, this thesis establishes a virtualisation system of FPGA resources which will be introduced in Chapter 4. The disaggregation of FPGA resources is then reviewed. It can be classified into vertical and horizontal systems. Vertical systems aim to develop a centralised resource manager that abstracts and simplifies the configuration (including updated registers and download bitstreams). The horizontal system attempts to develop boundless links between each FPGA device. In Chapter 5, in lieu of vertical configuration, a network protocol and associated processor that delivers horizontal configuration with a low downtime is developed.

CHAPTER 3

HIGH PERFORMANCE NETWORK INTERFACE FOR RESOURCE DISAGGREGATION

3.1 Introduction

In response to the growing data centre workload, in addition to buying more racks and servers, improving the data centre architecture is also suggested. For example, instead of CPU-centric architecture, the Hewlett Packard company suggests server racks with massive stores of dense, energy-efficient memory which are dedicated to providing storage services. To efficiently make use of data centre resources, resource disaggregation is also suggested in the dRedBox project [66].

The greatest burden of the scale-up of data centres falls on the network. Current data centres may consist of hundreds of thousands of servers arranged in a hierarchical tree-based structure [67]. All servers should be able to communicate with others through high-performance connections. Recent research findings suggest that over 70% of data exchange traffic remains within the data centre for intra-data centre communications [68].

Furthermore, new data centre architectures also have a strong requirement with the network. As [69] suggests, servers with high-density computing resource occupy more capacity in the data centre network. A high-performance network also acts as a key enabler of data centre resource disaggregation, as communication that was previously contained within a server is now provided by the external network [70].

Optical networks have a high potential to meet the requirements of recent large-scale data centres or new data centre architectures. Although fibre consumes much less energy than copper cable, high-bandwidth optical transceivers remain power-hungry modules on the server [32]. Current commercial off-the-shelf servers only include one or two SFP+ cages for a 10-Gbps interface. However, advanced data centre architecture requires a more powerful interface to establish high-bandwidth and low-latency connections between a large number of endpoints to support newer data centre architecture.

Optical transceivers with advanced photonic technologies have been invented to support high-capacity networks, efficient energy consumption, and coordination with fast electronic devices. In this chapter, a prototype network interface is developed to support data centre resource disaggregation. The contribution of this chapter is outlined as follows:

- Under the new network interface, recent optical transceivers based on silicon photonic technology are combined with FPGA devices.
- Related equipment, experiment, and skills which are used to combine the optical transceiver with the FPGA are introduced.
- By connecting the new implemented network interface with the optical switch, the prototype of a pure optical data center network is established.
- Based on the prototype of the pure optical network, the feasibility of data centre resource disaggregation is also evaluated.

The rest of this chapter is organized as follows. Section 3.2 introduces the motivation for using advanced optical transceivers in the developed network interface. Section 3.3 introduces the experiment to combine the transceiver with the FPGA and build the network interface. Section 3.3 introduces the experiment set-up. Section 3.4 presents the result of the experiment and analyses the feasibility of resource disaggregation. Section 3.6 provides a conclusion for this chapter.

3.2 Motivation

Two elements of optical transceivers can physically limit the bandwidth of a server's network interface: energy consumption and footprint dimension.

The data centre needs a large amount of power to function. Google reports that they continuously draw 260 megawatts of power to run their data centres [71]. Energy efficiency becomes a significant parameter of a data centre due to its economic and performance impact. Power bills become the most important expense of a data centre. On the other hand, techniques that reduce electricity usage of data centres also tend to reduce performance. As such, it points to a complex trade-off between energy consumption and performance. More optical transceivers are often included in the servers to support higher bandwidths. However, more optical transceivers integrated into the servers increase the servers' energy consumption.

Another restriction of optical transceivers involves the dimension of the footprint. Recent silicon fabrication with nanometre-scale accuracy has the ability to integrate large amounts of computing resources into the server. An increasing number of optical transceivers are necessary to support high bandwidth inter-server communication between these resources. The footprint dimension of the transceiver becomes important in order to include high-density optical transceivers on the server board.

The standards for 40/100-Gbps optical transceivers are CFP4 and QSFP28, both of which, have already been commercialised. In addition to standard transceivers, optical transceivers based on advanced silicon-photonic technology have also been invented. In this chapter, the OptoPHY transceiver is applied and analysed.

Table 3.1: The resource utilisation of the protocol processor

Module	QSFP28	CFP4	OptoPHY
Line rate (Gbps)	4x25	4x25	8x25
Hot pluggable	✓	✓	×
Dimension (mm)	18x52x8.5	22x92x10	36x28x12.8
BW volume (Mbps/mm^3)	12.6	4.9	15.5
Power (W)	3.5	5	5.6
Energy efficiency (pj/bit)	35	50	23

The parameters of the CFP4, QSFP28, and OptoPHY are listed in Table 3.1. The bandwidth volume and energy efficiency of the three optical transceivers have also been calculated. For the bandwidth volume, the OptoPHY is able to support higher bandwidths using a smaller volume. The OptoPHY also has better energy efficiency. The only limitation of the OptoPHY is that it is not hot-pluggable. The OptoPHY should be fixed and mounted on the board, whereas the CFP4 and the QSFP28 have a standard cage on the board to support the hot-pluggable feature.

Therefore, this chapter establishes a prototype network interface by combining the OptoPHY transceiver with an FPGA device. With the implemented network interface, this chapter aims to test whether new optical products equipped with advanced silicon photonic technology are able to support new data centre architectures and evaluate the feasibility of data centre resource disaggregation.

3.3 Experiment

This section introduces the testbed and experiment. Specifically, the prototype network interface is introduced first. The OptoPHY and FPGA are set up in the prototype. For comparison, the SFP+-based platform is also established. Then the related technology and skills needed to build the network interface and measure the results are introduced.

3.3.1 Testbed set up

Figure 3.1 illustrates the setup of the experiment. The FPGA is used to emulate the network interface of the end server. The gigabit transceiver (GT) module in the Xilinx IBERT needs a reference clock. Thus, another board with a Si5341 chip is used to provide a reference clock to the FPGA (200MHz and 312.5MHz for 16Gbps and 25Gbps, respectively). The Xilinx IBERT is deployed in the FPGA to measure the bit error rate. The OptoPHY optical transceiver is set up in the experiment to enable communication between the FPGA and the optical network. It is controlled by the Luxtera Control Suite. The power that is received can be monitored through the Luxtera Control Suite, and the clock data recovery (CDR) on the OptoPHY can be enabled here. An optical switch is also established to act as the optical data centre network. The port map of the optical switch is controlled through a serial interface from any terminal tools (such as PuTTY or TeraTerm). A network traffic generator is also connected to the optical switch in order that more diverse traffic can be sent to the system.

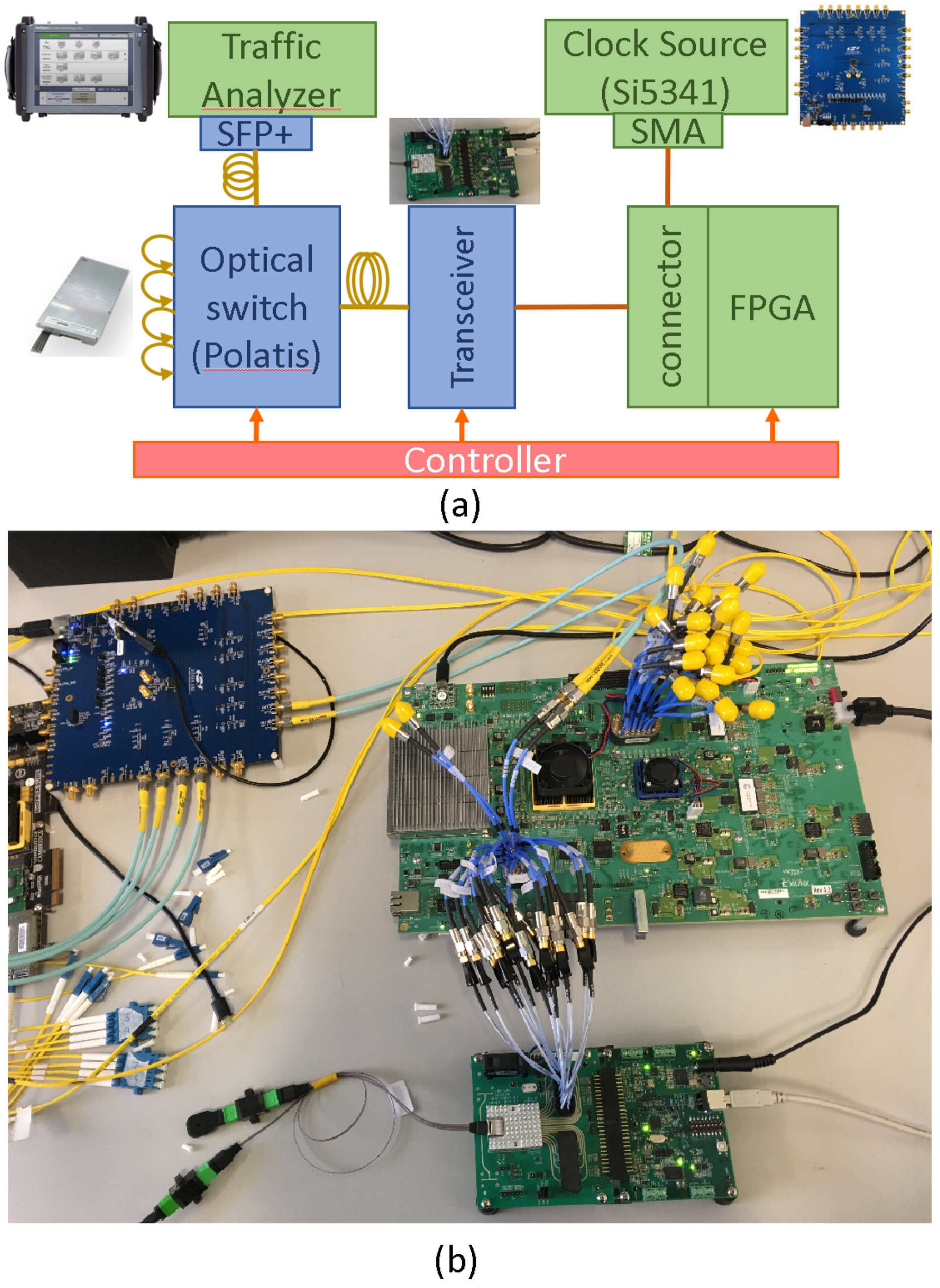

FIGURE 3.1. The testbed set up of the OptoPHY transceiver. (a) The structure of the testbed. (b) the Photo of the testbed.

The SFP+ is a common optical transceiver in current data centre servers. Therefore this chapter sets up the SFP+ to make a comparison with the OptoPHY transceiver. As shown in Figure 3.2, a Polatis 48-port single-sided all-optical switch module was established as the optical backplane. A NetFPGA SUME with low-cost SFP+ (LRC 10km, 1310nm) has been set up to serve as the traffic analyser. The Xilinx IBERT has been configured on the FPGA to collect the results. The bit error rate (BER) eye diagrams at different locations were also collected.

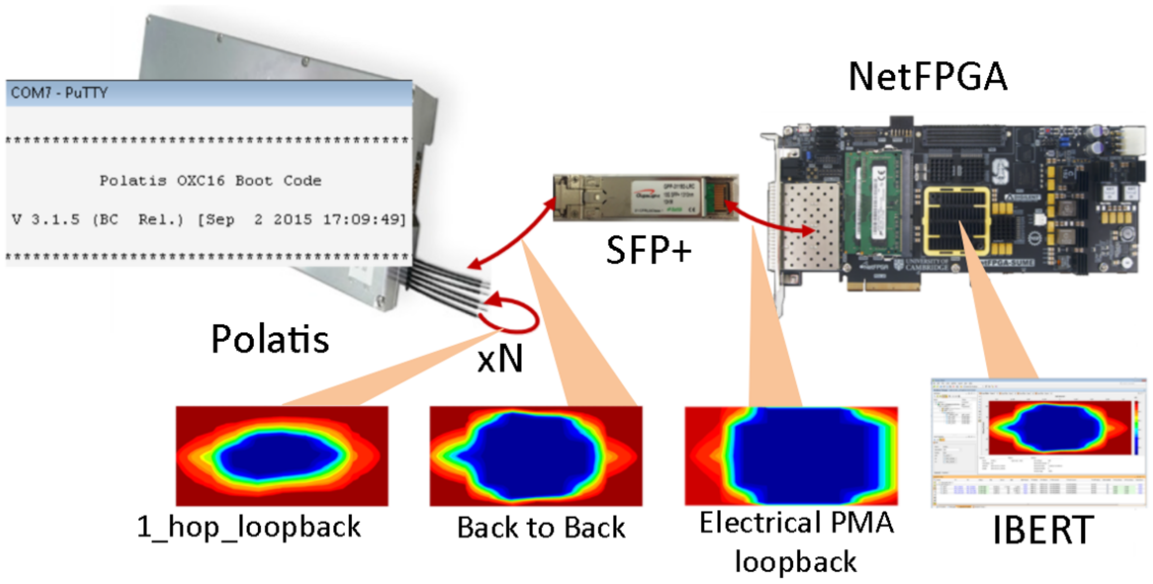

FIGURE 3.2. The testbed set up of the SFP+ transceiver.

3.3.2 Technology and skill

The feasibility of combining the OptoPHY with servers is evaluated by testing the quality of the optical link. This part introduces tools that can test the quality of the link from the FPGA chip. The Xilinx IBERT is an IP core that measures the BER of on-chip transceivers to test the quality of a physical link. Utilising the Xilinx IBERT often involves the following steps:

1. Decide the frequency of the reference clock
2. Choose the number of GT channels
3. Set the PIN numbers of the GT channels and the reference clock
4. Test the IBERT independently and test the link
5. Read the eye scan diagram

3.3.2.1 Decide the frequency of the reference clock

The frequency of the reference clock must be set during the initial design of the Xilinx IBERT, before synthesis. The frequency of the reference clock to be set should be the same as the frequency of the clock source to be used. The Xilinx gigabit transceiver often needs an independent reference clock. Moreover, it does not use the general system clock. However, just like the general system clock, the reference clock also comes from the PIN of the FPGA chip. The PIN for reference clock can be connected with an on-board clock source or with a clock source from the clock generator outside the board. There is no requirement on the frequency of the external clock, but it should be connected through the SMA or Bullseye connector and follow the LVDS standard. For a 10 Gbps 64bit system, the reference clock will be 156.25 MHz.

3.3.2.2 Choose the number of GT channels

The number of GT channels should also be set prior to synthesis. The GT channels are used to connect peripherals to FPGAs through serial interfaces. Each channel is connected with a pair of differential ports of the chip. The maximum data rate depends on the type of GT (e.g. GTX, GTH, GTY). The required number of the GT channels depends on the connected peripheral. For example, SFP+ needs a 10-Gbps interface; thus, it only needs one GT channel. The OptoPHY can support a maximum of 8x25 Gbps; thus, it needs 8 GTY channels to be able to run at its maximum performance. When both the frequencies of the reference clock and the number of GT channels are set, the Xilinx IBERT can be synthesised.

3.3.2.3 Set the PIN numbers of the GT channels and the reference clock

Like the design of other FPGA projects, the PIN of the Xilinx IBERT on the chip should be located after synthesis. The allocation of GT channels is the same as the allocation of other PINS. By finding the GT section in the user guide or schematic of the target board, the connection between the GT channel (a pair of differential pins) and the peripherals will be listed. However, allocating the reference clock is complicated because the clock structure of the GT itself is complicated. Figure 3.3 illustrates the clock structure of the GTs. Every four GT channels are physically located together as a group or QUAD in Xilinx FPGA chips. A quad of channels (four channels) shares two pairs of differential reference clocks (GTREFCLK0, GTREFCLK1 in Figure 3.3). In 7 series, each quad of GT channels can also use reference clocks from its physically-adjacent QUADs (GTNORTHREFCLK, GTSOUTHREFCLK). The PIN number of the reference clock of each QUAD can be found in the GT section of the user guide or the schematic of the board. There is no need to set the value of QPLLREFCLKSEL in Figure 3.3 when using GTs because this value will automatically be set according to the PIN location of the reference clock in the routing state of implementation.

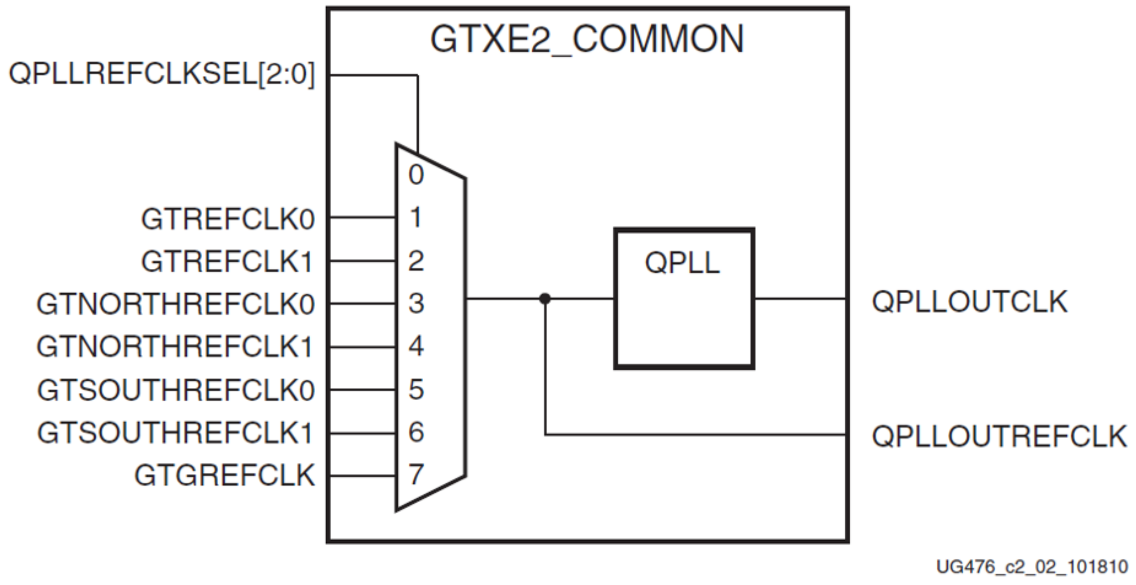

FIGURE 3.3. The clock structure of the Xilinx gigabit transceiver.

3.3.2.4 Test the GT module and the link

After the implementation and bit file generation, the Xilinx IBERT can be performed to test GT modules and the physical or optical links. Some options are available in the IBERT which help users validate the link and collect results.

There are two types of loopback tests: near-end test, and far-end test as shown in Figure 3.4. The GT channels can be tested by internal monitors in the near-end test. An external analyser can also be used to test the channels. The loopback mode can be selected in the hardware manager of the Xilinx Vivado.

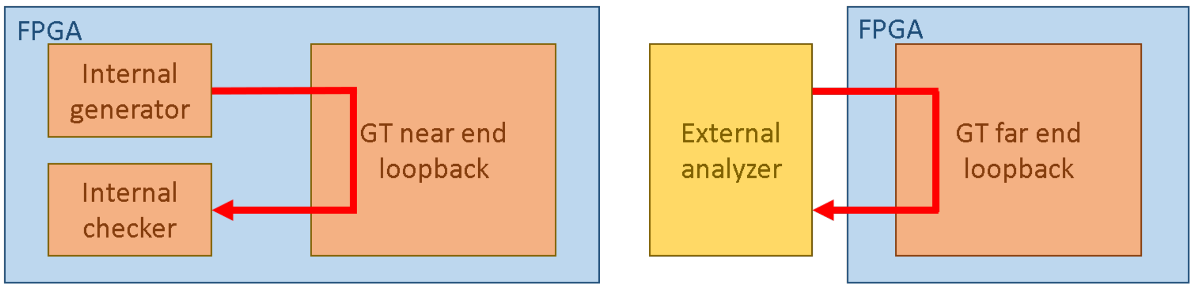

FIGURE 3.4. The loopback mode in the Xilinx IBERT. (a)Near end loopback. (b)Far end loopback.

In addition to the BER of the physical link, the IBERT can report more detailed information about the link in the eye diagram. A typical eye diagram collected by the Xilinx IBERT is shown in Figure 3.5. The Y-axis of the diagram represents the voltage, while the X-axis represents the time within a unit interval (the time during a period of cycle of the received signal). The point of origin (0,0) is located in the middle of the diagram. The bit error rate of the origin point will be counted as the BER of the received signal in the Xilinx IBERT.

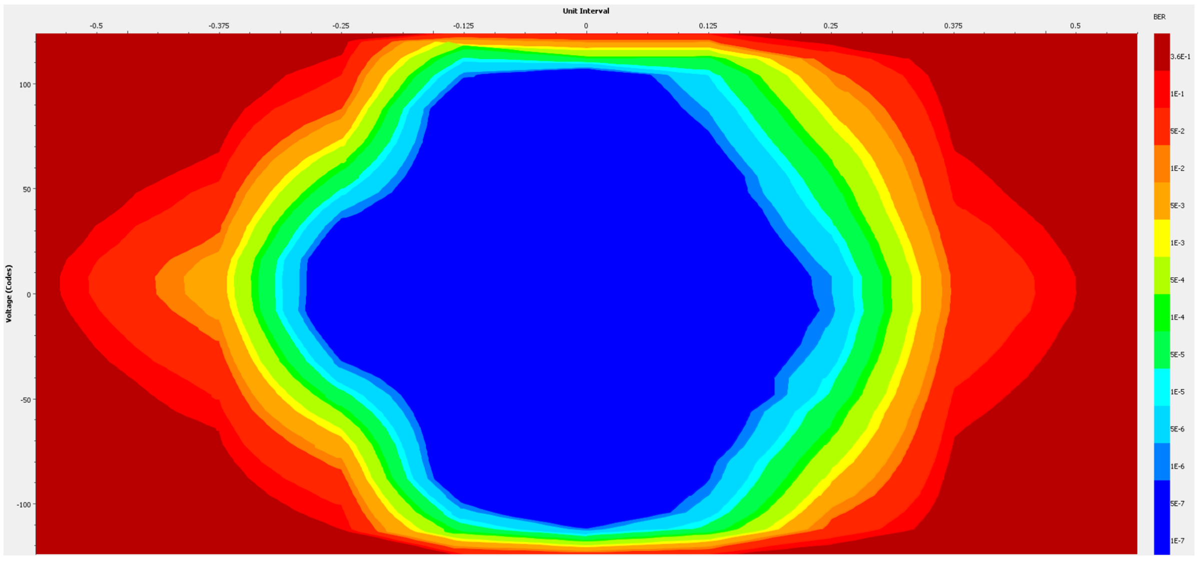

FIGURE 3.5. A typical eye diagram collected by the IBERT.

The bit error rate of other points can also be collected and be shown in the eye diagram. The

IBERT is able to move left or right of the sample point by sampling the received signal earlier or later in the cycle. The IBERT can also move the sample point up or down by sampling the received signal in a higher or lower voltage. The bit error rate of all the sample points can be collected as an array of values. This array is converted into figures and is shown in the eye diagram. Blue means a lower bit error rate value and red means a higher value. A large blue area in the middle means the physical link is reliable. Otherwise, the physical link can easily be affected by unknown random events in the environment.

3.4 Result

This section demonstrates the results of the experiment. In order to meet the performance requirement of data centre services, optical networks should be able to deliver high-quality signals. Thus, the bit error rate of the OptoPHY transceiver is first collected and compared with the traditional SFP+. To support advanced data centre architecture such as resource disaggregation, complex network topology must be supported. Therefore, this chapter evaluates the scalability of the prototype network interface. The network latency of Ethernet layers is also measured to analyse the performance of higher layers.

3.4.1 Optical transceiver result

The OptoPHY runs at 16Gbps and 25Gbps respectively, and through a number of optical switch cross-connections. The bit error rate of SFP+ is also collected for comparison. As shown in Figure 3.6, the OptoPHY transceiver at 16 Gbps is able to deliver a 2-dBm improvement compared to SFP+ while achieving error-free (1E-12-bit error rate). Also, when the clock data recovery is enabled at 25 Gbps, the OptoPHY can offer an improvement of approximately 1 dBm.

3.4.2 Optical switch result

The established network interface enables direct high bandwidth connections between servers and optical networks. To evaluate the performance of the overall network, a prototype network that combines the new network interface with optical switches is set up. The power and the BER

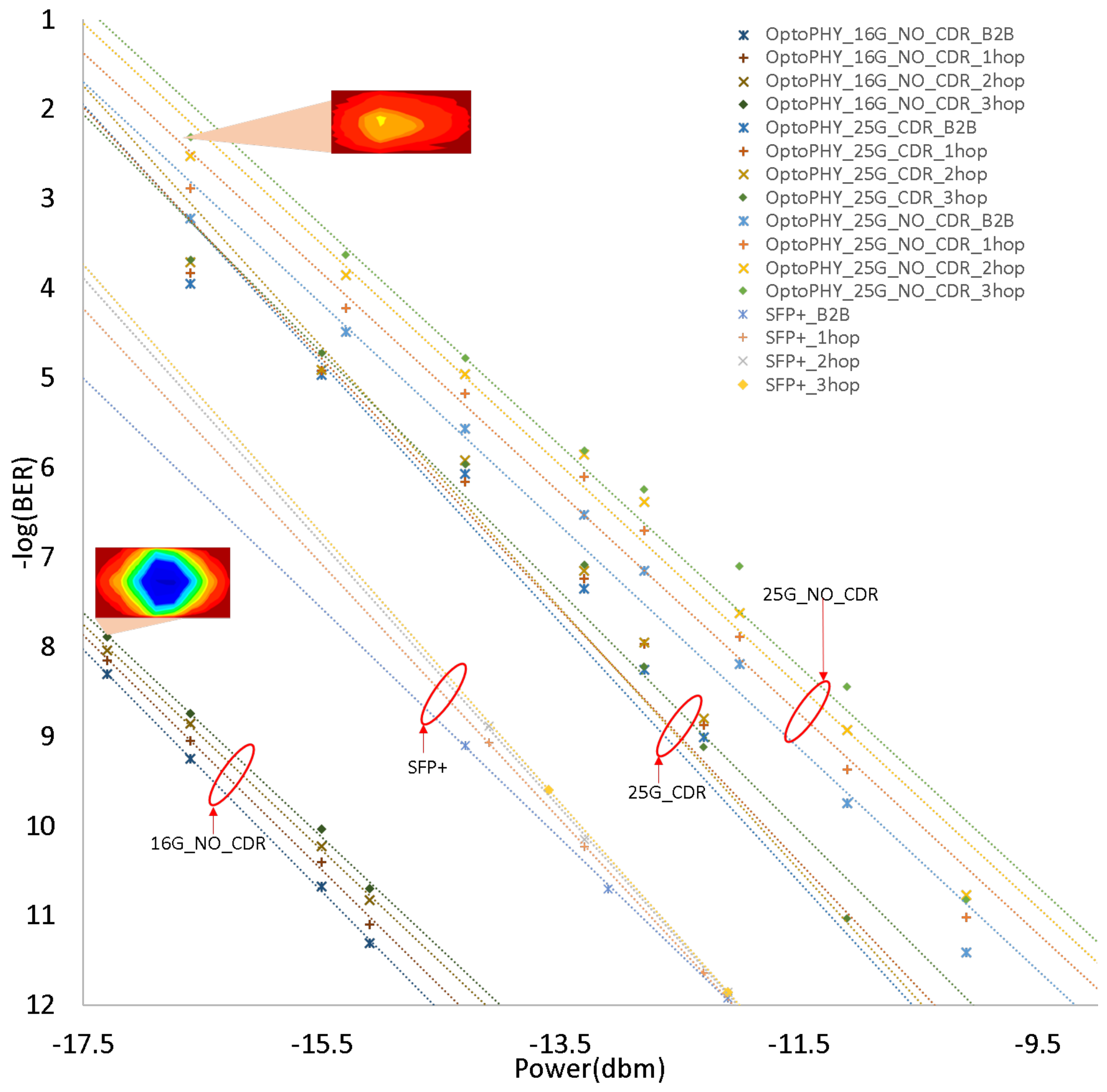

FIGURE 3.6. The bit error rate (BER) of the OptoPHY and SFP+ optical transceiver. Lines that closer to the bottom-left have better BER.

of the received signal are measured as shown in Figure 3.7. The optical switch introduces an average power loss of approximately one dBm per hop. The SFP+ can support a maximum of four hops while the signal sent by the OptoPHY can still be detected at the seventh hop.

This chapter evaluates the scalability of the proposed optical data centre network based on the result of the optical switch. The relationship between the bit error rate and the number of

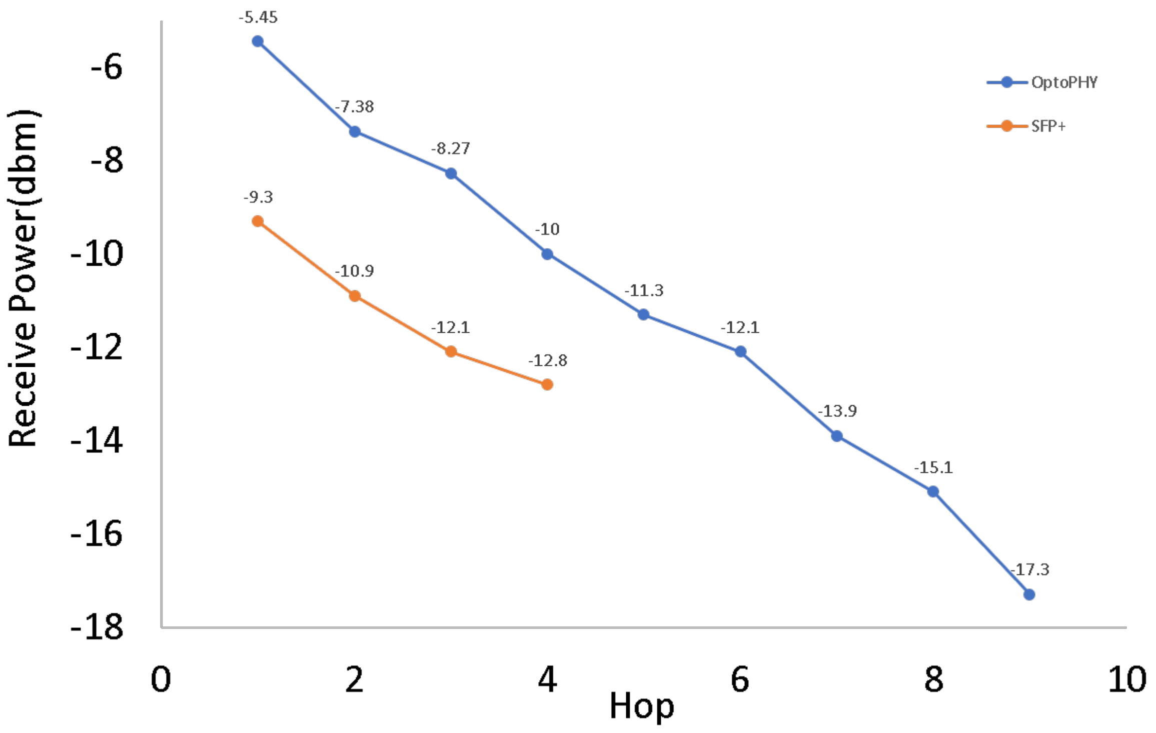

FIGURE 3.7. The loss of power of the optical switch.

hops is shown in Figure 3.8. In the evaluation, the bit error rate 1E-12 is adopted as the error-free threshold because bit error rates lower than 1E-12 do not need frame error corrections in the Ethernet system (which introduces hundreds of nanoseconds of latency).

The SFP+ transceiver can offer an error-free signal over two hops of the optical switch. It corresponds to 94 end points as shown in Figure 3.9. The OptoPHY at a 16-Gbps transceiver can support seven hops of error-free optical links, which corresponds to a 5-tier network with more than 5,000,000 end points as shown in Figure 3.9. Furthermore, the OptoPHY running at 25 Gbps can still deliver four hops to support a two-tier network with more than 2,300 end points as shown in Figure 3.9. The results indicate that OptoPHY is able to offer high scalability to data centre servers compared to the traditional SFP+.

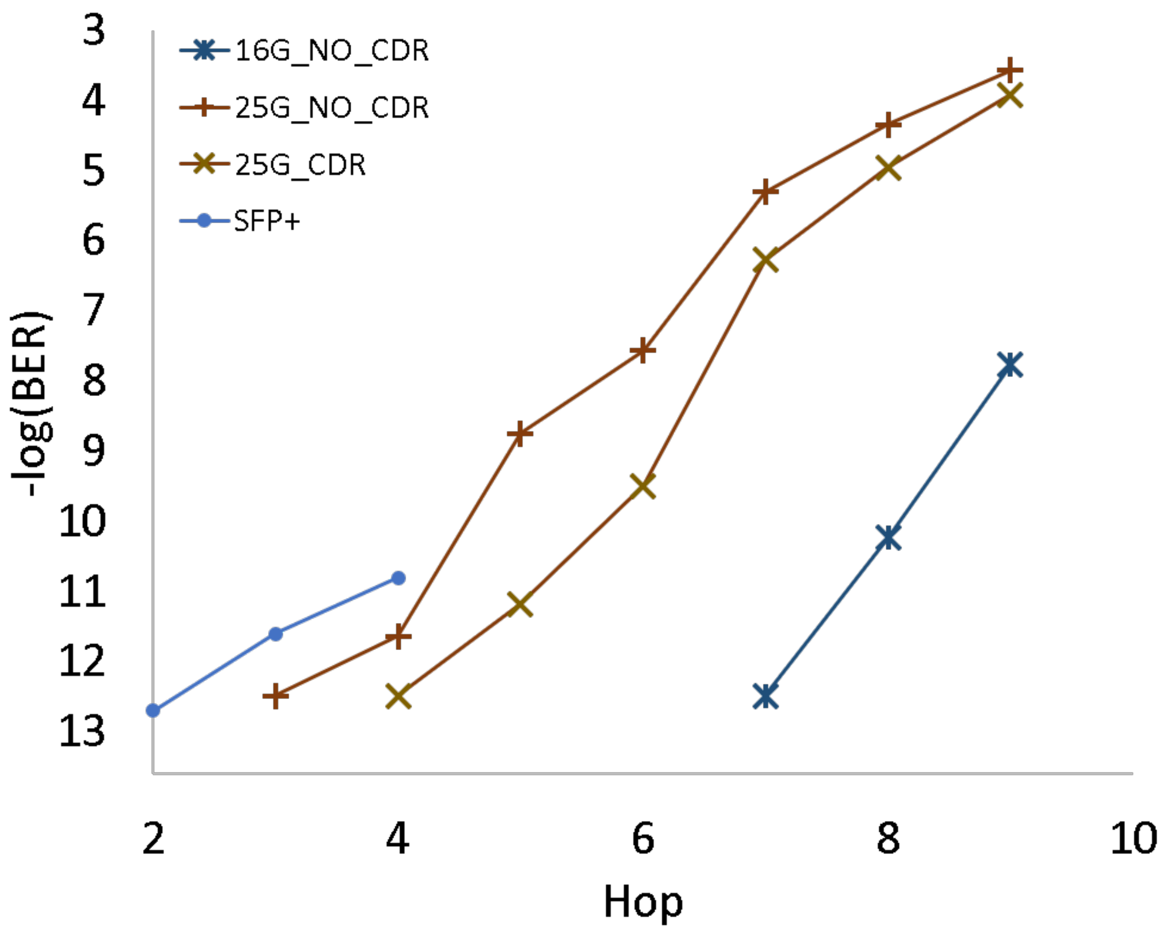

FIGURE 3.8. The relationship between the hops and the BER.

3.4.3 Ethernet layer result

This chapter also builds a standard 10G-Ethernet interface by combining the OptoPHY with FPGA devices, and a 10G-Ethernet switch is also built based on the implemented 10G-Ethernet interface. As shown in Figure 3.10, the latencies of both the OptoPHY and the SFP+ transceiver are measured. The network interfaces built with SFP+ and OptoPHY have similar latencies.

3.5 Limitation

Although the experiment and research were carefully prepared, limitations and shortcomings still need to be mentioned.

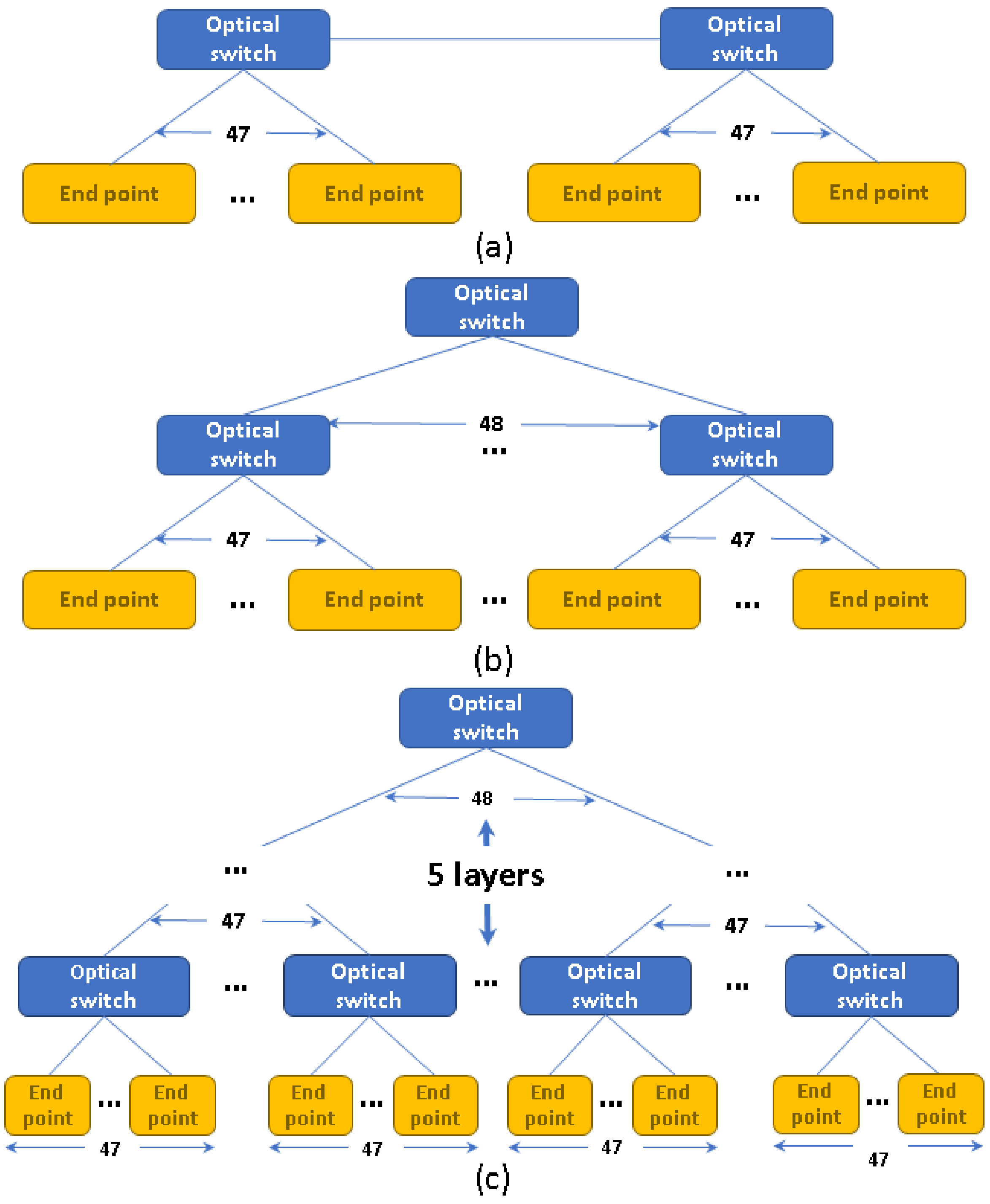

FIGURE 3.9. The scalability of a multi-tier network. (a) A tier 1 network can support around 100 end points. (b) A tier 2 network can support around 2300 end points. (c) A tier 5 network can support more than 5,000,000 end points.

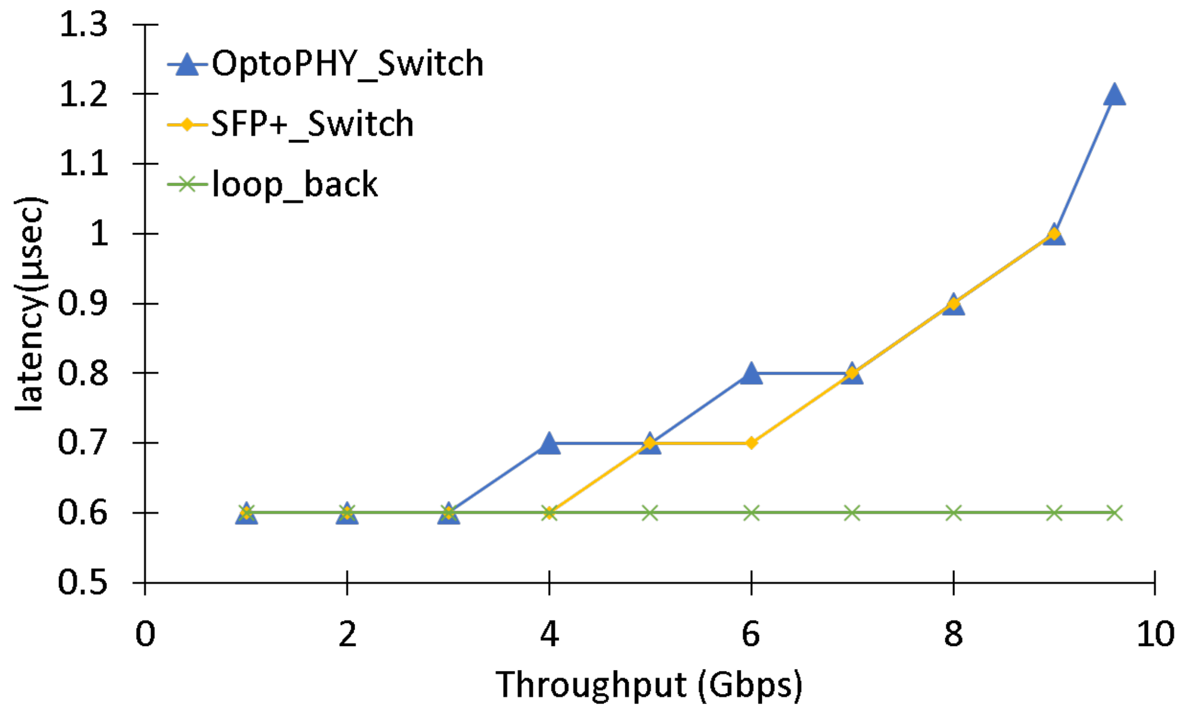

FIGURE 3.10. The measured latency at the Ethernet layer.

First, in Chapter 3, the experiments for the SFP+ optical transceiver and OptoPHY transceiver were conducted at different times and locations. The experiment for the SFP+ transceiver was conducted in August 2016 in Bristol, while the experiment for the OptoPHY was conducted in March 2017 in London. As environmental factors such as temperature may have an impact on the physical links, it may make a slight difference in the quality of the received signals.

3.5.1 Summary

In summary, OptoPHY transceivers are able to deliver a better signal quality than the SFP+s that are widely utilised in servers, and both kinds of transceivers have a similar layer 2 performance. With better quality, signals sent by OptoPHY are able to reach more endpoints after more hops. It is able to significantly increase the scalability of servers and make resource disaggregation possible.

3.6 Conclusion

To support the rapidly-growing data centre workloads, new data centres require networks with higher bandwidth and scalability. This chapter demonstrates a prototype network interface equipped with optical transceivers based on advanced silicon photonic technology. The proposed network interface is able to offer high-bandwidth density to servers. Related technologies to set up the network interface are introduced. This chapter has also combined the network interface with an optical switches to evaluated the feasibility of resource disaggregation. The result has shown that recent advanced optical transceivers and switches are able to support multi-tier networks with a large numbers of endpoints.

CHAPTER 4

THE VIRTUALISATION OF RECONFIGURABLE FPGA RESOURCES

4.1 Introduction

Due to their flexible and parallel processing features, the reconfigurable resources act as flexible accelerators that improve the performance of data centres. To apply the reconfigurable resources in the data centre environment, the virtualisation of reconfigurable resources is required. To conveniently share reconfigurable resources to users, they should properly be isolated and partitioned. Therefore, this chapter explores architectures that virtualise reconfigurable resources.

On the other hand, with the development of computer networks, people find it challenging to deploy new network technologies, services, and protocols in the internet environment as the current internet infrastructure is hard to manage, upgrade, and evolve. In this situation, the software-defined network (SDN) is proposed to explore innovations in network design and operations [72].

The SDN proposes to centralise network control and introduce the ability to program the network. As the network is defined by software applications that are designed and executed in operating systems running on general purpose processors, the capital and operational expenditures to manage and upgrade networks can be reduced. As a complement to the SDN, network function virtualisation (NFV), instead of running network applications on real operating systems, suggests running these applications on virtual machines [73]. Thus, the hardware resources of the network can be isolated and shared.

Although running software network applications on operating systems increases the flexibility and programmability of network services, general-purpose processors can limit the bandwidth of these network applications, especially on data planes. To maintain both the programmability and bandwidth of network services, the FPGA being a reconfigurable device is another outstanding reconfigurable hardware platform for NFV, especially in the virtualisation of network data planes.

FPGA is also a type of useful resource in data center. There are direct feedbacks from the commercial market: comparing to CPU, FPGA can improve the performance significantly, while comparing to GPU, FPGA need much less power. These advantages of FPGA help to reduce the operational cost of a data center. There are also more potential benefits which have not been seen from the commercial market. For example, it would be very easy and fast to upgrade high performance functions to newest version, prototype cloud data processing can be test and validated first on the FPGA resources.

To keep the advantage of NFV that shares hardware resources with multiple network services, the FPGA hardware platform needs to be virtualized before delivering network functions. However, the virtualisation of FPGA resources is not as mature as the virtualisation of general-purpose processors.

This chapter explores the virtualisation technology of FPGA resources to deliver high-bandwidth virtual network functions. An NFV platform is implemented in this chapter to deliver network data plane stream processing on virtualized FPGA resources. The contribution of this chapter includes the following:

- Architecture that virtualises FPGA resources is implemented. In the proposed architecture, the technology of partial reconfiguration and on-chip interconnect is applied to isolate FPGA resources. FPGA resources can also be directly attached to the 10G Ethernet without involving processing units in the network data plane.

- The on-chip interconnect system in the proposed architecture is evaluated and compared with the existing interconnect system.

- A reconfiguration process that is able to switch partial reconfigurable network functions in real time and in a packet loss-free manner is proposed and demonstrated.

The rest of the chapter is organized as follows. First, Section 4.2 introduces the background. Section 4.3 provides and explains the overview architecture of the proposed platform for reconfigurable network stream processing. Section 4.4 evaluates the on-chip interconnect in the proposed system. Section 4.5 compares the proposed reconfiguration process with a typical partial reconfiguration process. Section 4.6 demonstrates the experiment to validate the platform and demonstrate the proposed reconfiguration process. Finally, Section 4.7 provides concluding remarks for this chapter.

4.2 Background

The SDN is a network paradigm that aims to decouple the control plane and data plane of the network. The SDN enables centralised control and management on the distributed data plane of a network [74]. The SDN also suggests a standard software control plane which can be independent of its hardware [33]. Although the SDN only suggests a software control plane, a software data plane is often adopted as a complement of the SDN concept. Especially, in data centres, software network switches are often used between virtual machines that are deployed on the same server [75].

These software applications on network control and data plane enlighten the concept of an NFV. Network functions are no longer treated as functions of an ASIC-based hardware platform built with specific purpose. Instead, they are now software applications that run on the operating system of general purpose processors. Therefore, virtualisation technology in computer systems can be applied to isolate and share network resources with multiple tenants [76]. The NFV first proposed to virtualise the control plane by using SDN technology. Then, both the control and data planes of a network function are virtualised to deliver various network services such as firewalls and load balancers. Furthermore, services beyond the OSI application layer (such as video processing services) are suggested to complement NFV especially in a data centre environment [77]. The NFV inherited many of the SDN's features and advantages. For example, network services can be fast and easily deployed by using NFV technology [78]. The network can be adjusted based on real-time traffic and the requirements of network services. Moreover, NFV also has its unique advantages such as its ability to share physical network resources with multiple tenants [76].

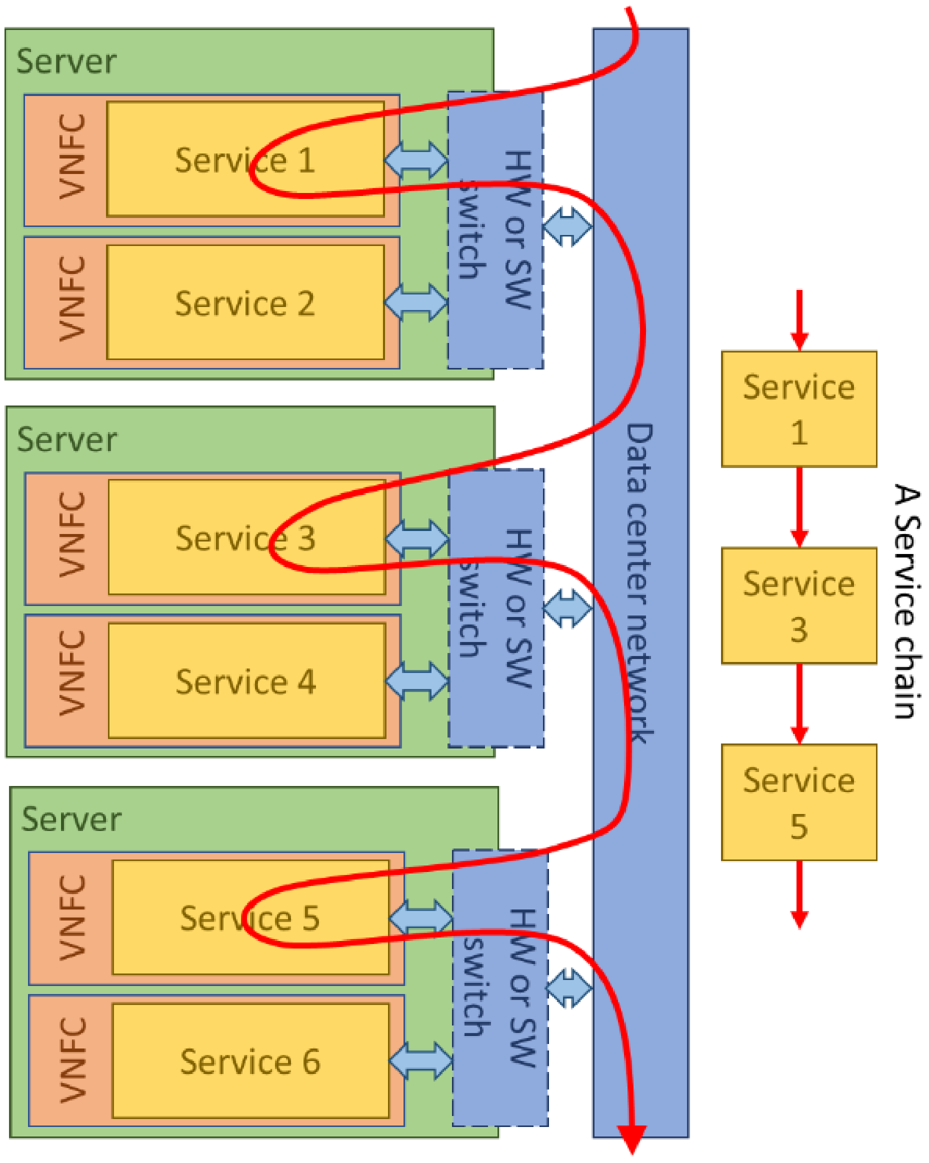

FIGURE 4.1. A typical infrastructure of NFV and the service chain.

Figure 4.1 shows a typical NFV infrastructure in data centres. Servers are the common hardware platform of NFV in data centres [79]. The software that enables the virtualisation of servers is called a hypervisor (also known as virtual machine monitor) [80]. The hypervisor is responsible for isolating and allocating physical resources on the computing platform to individual virtual machines (virtual network function containers). The hypervisor in NFV is often specialised for network virtualisation. For example, the hypervisor is often equipped with a hardware or software network switch. Services deployed on virtual machines can be connected based on the demand of users to build a chain of services as marked in the red line of Figure 4.1 [81].

Although moving network functions from hardware to software decouples functions from its physical platform, it reduces the performance of network functions. Therefore, additional hardware accelerators are often required by the network to meet performance requirements. Moving back to ASIC accelerators increases the coupling degree of control and data planes which go against the design philosophy of NFV. Hence, FPGA as a reconfigurable but high-performance platform is adopted [47]. As the processing on FPGA can be customized deep to RTL level, introducing FPGA to the NFV can give network developers a chance to trade off between the performance and the flexibility.

Similar to the NFV on general purpose processors, the NFV on FPGAs should also be built on the virtualisation technology of FPGAs to share physical resources with multiple tenants. Partial reconfiguration has become the key technology to enable the virtualisation of FPGA resources. [82] explores the integration of the partial bitstream download process with the operating system. This research enables the downloading of bitstream. However, the partial reconfigurable regions in this research is relatively closed enviroment, because the interfaces to off-chip network, to static region and between partial reconfigurable regions are not mentioned. The hardware environment of the FPGA virtualisation is implemented in [83]. Specifically, a PCIe interface has been set up as the interface between the processing unit and programmable logic. in this research, reconfigurable regions are interconnected through PCIe interfaces and the host machine. The PCIe interface introduce additional latency as the data have to go out of the chip to reach another reconfigurable region. Routing all the data to correct regions also introduce delays. [58] improves virtualisation by enhancing software support, especially for the data centre environment. Similar to previous researches, [58] access partial reconfigurable regions through a memory-mapped interface. And the communication between partial reconfigurable regions is done by the host processors. However, there are also routing resource inside the FPGA chip. The on chip routing resource introduce less delays to reach other partial reconfiguration regions. Also, it is possible to use stream processing on the partial reconfigurable regions instead of memory-mapped processing.

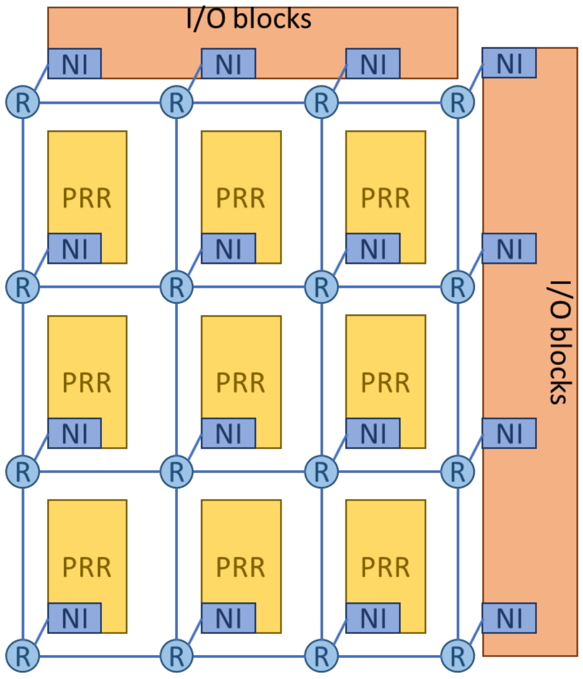

FIGURE 4.2. A typical NoC based architecture for FPGA vidtualisation.

Both the data and control planes are coordinated by the processing unit in aforementioned research work. However, to virtualise FPGA for network functions, involving general purpose processors in the data plane can lower performance, especially bandwidth. Therefore, instead of using processing units, FPGA-based interconnect architectures are taken into account in virtualising FPGA resources. The research by [53] proposes a network-on-chip based design to share FPGA resources. Research conducted by [84] extends the NoC-based FPGA virtualisation in cloud computing. A typical NoC-based FPGA virtualisation architecture is shown in Figure 4.2. It is often composed of several partial reconfigurable regions where accelerators can be deployed. Data communication between the deployed accelerators is transferred through an interconnect system. For I/Os to communicate with the off-chip network, they are often attached to the interconnect. In this chapter, an FPGA system that follows the concept of FPGA virtualization is implemented to deliver reconfigurable high-bandwidth network stream processing in the data plane of NFVs.

4.3 Architecture Overview

In this chapter, FPGA virtualisation aims to support NFV. In other words, the physical FPGA platform must be shared by multiple network services that come from many users. FPGA adoption is also incorporated to virtualise the network function on a data plane. Thus, the virtualised network functions should support a high bandwidth. Therefore, the virtualised FPGA platform should support the following features: a) network functions should be configured independently, b) network functions should have high-performance interfaces, and c) network functions should share the network interfaces of FPGA boards.

The typical configuration of the FPGA needs to reprogram the entire FPGA device. However, when multiple network functions share an FPGA, reprogramming one of the functions requires all the other functions on the board to be switched off, which affects functions that belong to other services and users. Therefore, partial reconfiguration technology is adopted. It partitions the FPGA into one static region and several dynamic partial reconfigurable regions. The program of partial reconfigurable regions will not affect the function in the static region and other partial reconfigurable regions.

The design features of partial reconfiguration require a static interface between the dynamic and static regions. Hence, all the deployed network functions should be attached with a standard interface to enable the partial reconfiguration technology. The standard AXI4-stream is utilised as the interface in this chapter. An on-chip interconnect system is also implemented in the static region to support high-bandwidth network functions and establish data communication between dynamic regions.

FPGA devices often include a limited number of network interfaces. Thus, the network interfaces should also be shared by network functions when an FPGA platform is built to support NFV. Therefore, the 10-Gbps Ethernet interface is attached to the interconnect and makes the interconnect forward data at the Ethernet packet level. Thus, all the dynamic regions are able to gain access to the off-chip network through the interconnect.

The structure of the proposed FPGA virtualisation for NFV is suggested in [85] and is illustrated in Figure 4.3. It includes a number of partial reconfigurable regions (PRR) and an interconnect system. The 10-Gbps Ethernet systems are also attached to the interconnect system to directly transfer data through the off-chip network. A controller is also included to control the interconnect system and orchestrate the reconfiguration process of virtual functions (to be introduced in Section 4.5). All the modules run at 156.25 MHz, which is the frequency of the 10G Ethernet system with a 64-bit data width.

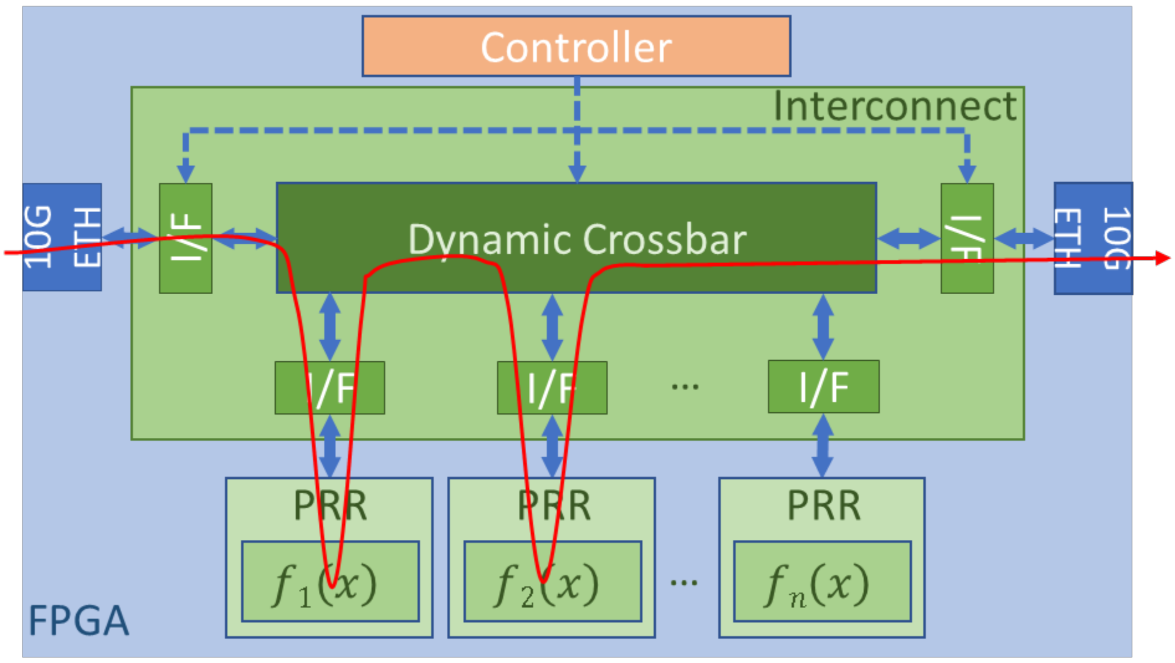

FIGURE 4.3. The overview of the proposed FPGA virtualization architecture for NFV platform.

A dynamic interconnect system is implemented and it forwards data between PRRs in the proposed platform. It is composed of a dynamic crossbar and interfaces to PRRs. The data are forwarded based on the local address signal in parallel with the data signal (as the USER channel in an AXI4-stream standard). Each PRR is identified using a unique address. The local destination address of a packet is added to the interface, and the crossbar forwards packets based on the attached local address. The architecture of the crossbar and interface will be introduced further in next paragraph. The 10-Gbps network ports are also connected with the interconnect system. All PRRs are decoupled from the physical interfaces and are directly attached to the interconnect system as pluggable independently accessible regions. This enables reconfigurable processing on network streams. The whole FPGA virtualisation system is managed by the central controller. Updated information is transferred through the central controller to (a) set the value of memory-mapped control registers and (b) change the port map of the dynamic crossbar. The central controller is also responsible for performing the function reconfiguration process. The detailed information of that process is discussed in Section 4.5.

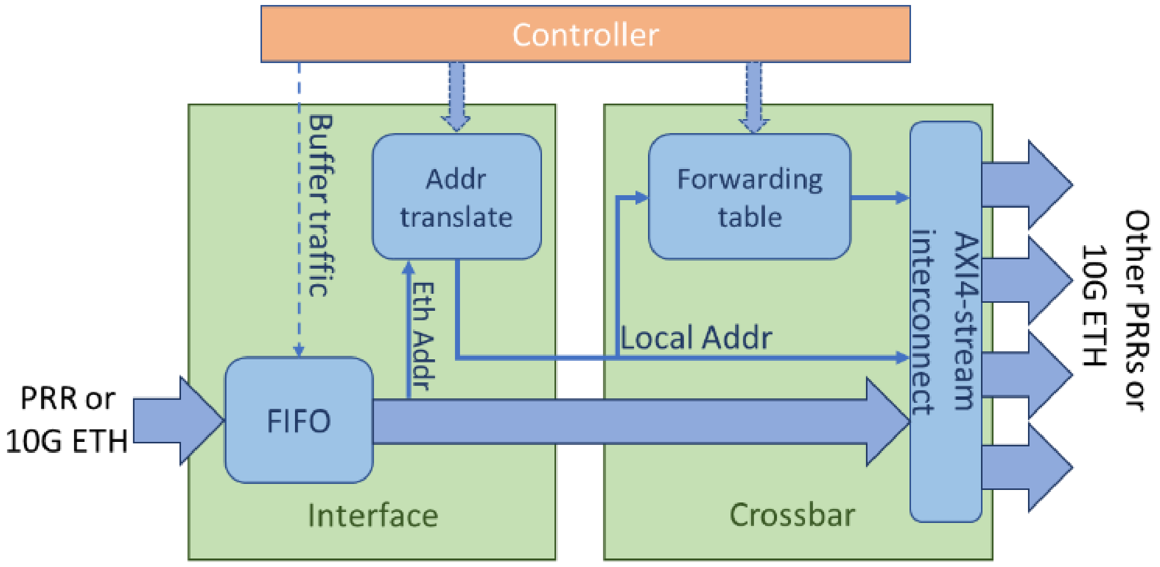

FIGURE 4.4. The detailed structure of the interface and dynamic crossbar of the interconnect.

The implementation details of the dynamic crossbar and interface are shown in Figure 4.4. The interface is located at the edge of the interconnect system to bridge PRRs and PHYs with the crossbar. The crossbar and interfaces are connected in accordance with the AXI4-stream handshake standards.

The Ethernet address is translated into a local address based on the table in the interface. This table can be updated from the controller through the AXI4-lite interface to enable real-time control on the interconnect system. The local address is transferred in parallel with the data (as the USER channel of AXI4-stream) until it reaches its required PRR. The interface is also responsible for buffering or releasing the traffic. FIFO is added to temporarily store the data while the reconfiguration of the interconnect goes on. The FIFO can be controlled from the central controller.

A forwarding table that stores the map information between the local address and output port is stored in the dynamic crossbar. When a packet is received by the crossbar, its local address (attached by the interface) is checked, and the packet is forwarded to the related output port by referring to the forwarding table. The forwarding table is dynamic and can be updated from the central controller using an AXI4-lite interface.

Because all network functions are only attached to the interconnect system and are clearly decoupled from the high-speed I/Os, the flexible FPGA virtualisation platform can deploy and involve network functions where and when they are required. A network stream processing unit can be reconfigured on PRRs on either a per port, per independent flow, or per virtual network basis. PRRs can also be reconfigured into requested functions at run-time without service disruptions or loss of data (introduced in Section 4.5).

Thus, a reconfigurable NFV platform is established with the following features: (a) Independent virtual FPGA resources or regions, (b) Shared network interfaces by network functions, (c) Network developers can enhance the already deployed functions in real-time and in a packet loss-free manner.

4.4 Evaluation of the interconnect

The interconnect system plays a significant role in the FPGA virtualisation system, as all the traffic received needs to go through the interconnect system before reaching its respective destinations. A low-performance interconnect structure will limit the performance of all the deployed network functions. Therefore, the interconnect system of the proposed FPGA virtualisation platform will be evaluated.

To put the resource utilisation and network performance of the interconnect system into perspective, the interconnect system is compared with the publicly available CONNECT [86] [87] and SOTA [88] network on chip. The SOTA mainly targets the ASIC design. When mapping to the FPGA platform, the SOTA needs a large number of resources. The CONNECT tries to solve this problem at the expense of reducing the maximum router frequencies [89]. In the following presentation, resource utilisation in the synthesis result of all the interconnect systems will be compared. Then, the cycle-accurate simulation results will be collected to evaluate the network performance. The performances of SOTA and CONNECT are reported in [86]. Furthermore, the simulation of the implemented interconnect system is done in the simulation tool in Xilinx Vivado.

Routers with 5 and 16 ports of all the three interconnect systems are set up. The data widths of both CONNECT and the interconnect system are 64-bit. The SOTA only supports a 32-bit data width. Additionally, resource utilisation figures in terms of LUTs are illustrated in the bar graph in Figure 4.5. The implemented interconnect system uses the least amount of resource among all the three interconnect systems.

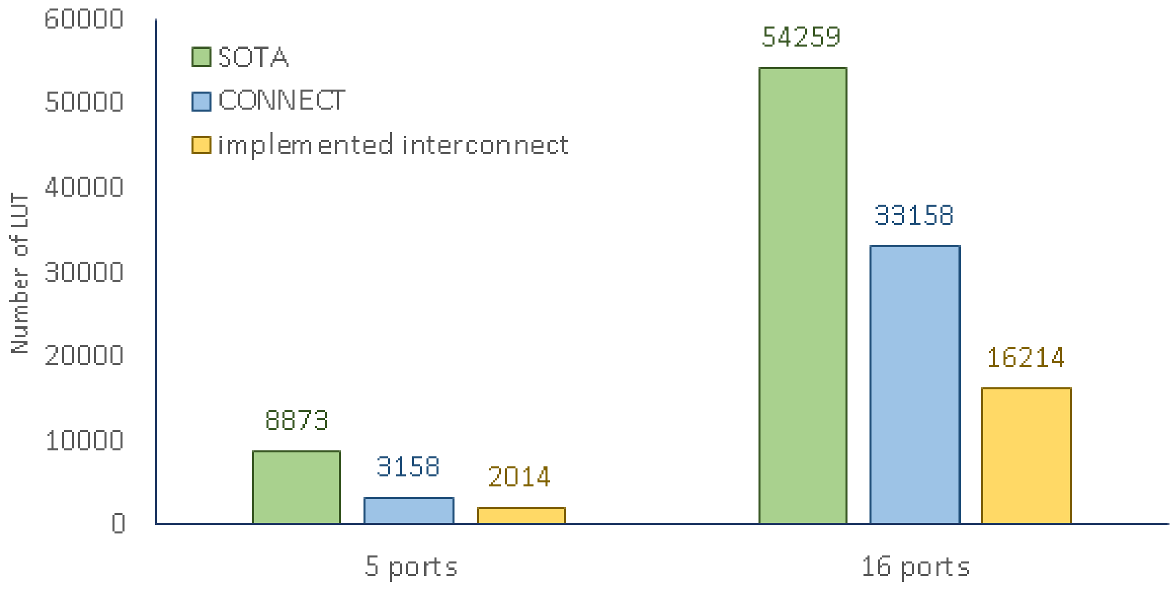

FIGURE 4.5. The LUT resource utilisation of CONNECT, SOTA and the interconnect system implemented in this chapter.

Two traffic patterns are set up, and the load-delay curves are measured to compare the network performance of the interconnect systems. The first traffic pattern is uniform random traffic, where the destination of each packet is randomly set. The second traffic pattern is hybrid traffic made of two patterns. 90% of the hybrid traffic is a static traffic pattern. The map of input and output ports of the generated flows is fixed as shown in Figure 4.6. Furthermore, the loads of the receiving ports are equal to each other. 10% of the hybrid traffic is uniform random traffic, which is the same as the first kind of traffic. The second kind of traffic depicts a traffic pattern where deployed functions heavily transfer data with several other functions in the system and also talk to the rest of the functions occasionally.

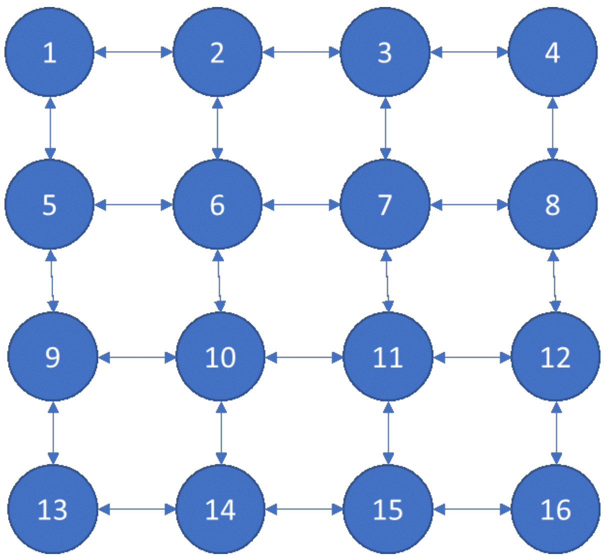

FIGURE 4.6. The pattern for the static part of the hybrid traffic.

Figure 4.7 shows the results. When all the three interconnect systems run at the same frequency, the interconnect system can support a maximum of 4.3 Gbps under uniform random traffic, and 5.8 Gbps under hybrid traffic, followed by the bandwidth of CONNECT with a 64-bit data width as shown in Figure 4.7 (a) and (b). However, the higher maximum bandwidth comes with delays. The interconnect system needs an additional 40 ns when there is a situation of random traffic, and 20-35 ns under hybrid traffic. The three interconnect systems are compared at their design frequencies. Under both traffic patterns, the delay of the interconnect system is similar to the 64-bit CONNECT. Nonetheless, the interconnect is still able to support higher bandwidths as shown in Figure 4.7 (c) and (d).

The features of the SOTA, CONNECT and the implemented interconnect are compared in 4.1.

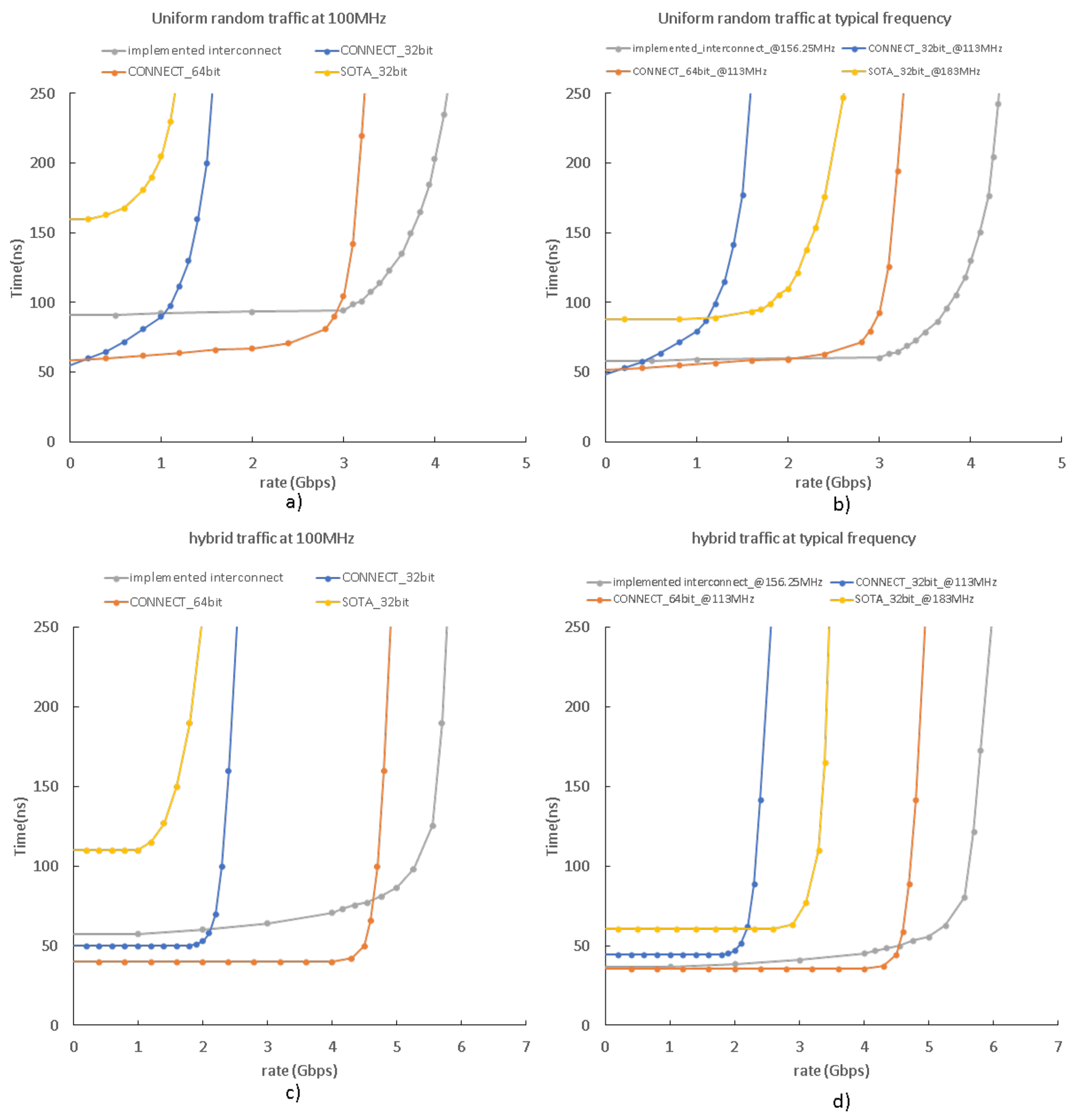

FIGURE 4.7. The comparison between the implemented interconnect system and the existing SOTA and CONNECT interconnect.

As the SOTA is designed for ASIC device, it has a fixed 32-bit data width., while the data width of the implemented interconnect and CONNECT are both variable. Both SOTA and CONNECT can support virtual channels, while the implemented interconnect system does not have the feature of virtual channel. The control information such as destination information is transferred in the header flit in SOTA, while CONNECT and the implemented interconnect system both provide dedicated wire for the control information.

Table 4.1: The comparison between SOTA, CONNECT and the implemented interconnect system

Interconnect	SOTA	CONNECT	Implemented
Data width (bit)	32	variable	variable
Virtual channel	✓	✓	×
Control information	in header flit	in parallel with data	in parallel with data

The SOTA system has the lowest performance under all circumstances. This is because SOTA only supports a 32-bit data width, which severely limits its performance. Unlike the interconnect system and the CONNECT system, in the SOTA system the control information including address is transferred in band together with the DATA. That also introduces overheads. The CONNECT system has a lower latency, especially when the system is not heavily loaded. This is because CONNECT is designed for computing systems. Delays are more important than bandwidth. The feature of virtual channel in CONNECT is able to reduce the delay. But arbitrating virtual channel introduces additional arbitrate clock cycles, which reduces the maximum bandwidth of the system. High bandwidths are important in a network system, so the virtual channel feature is not included in the implemented interconnect system. Other than bandwidth, 10G Ethernet interface and Ethernet address translation are also attached to make the system work more directly with off-chip networks.

4.5 Reconfiguration Process

The download process for a partial bitstream takes time when partial reconfiguration is performed. As the PRR cannot perform any function during this period, the adoption of partial reconfiguration can result in a loss of service. To avoid service loss, a function reconfiguration process supported by the system is suggested. This section introduces that reconfiguration process.

Instead of occupying all PRRs by network functions, a backup (empty) PRR is reserved only for the reconfiguration process. As the empty backup PRR does not receive and send any traffic, the bitstream for this backup PRR can be downloaded without disturbing existing traffic in the system. When a function reconfiguration (e.g. from IP parser to UDP parser) is requested, the partial bitstream of the requested function will first be loaded on the backup PRR. During the deployment of the partial bitstream on the backup PRR, the rest of the platform, including all the active PRRs, will neither be affected nor interrupted. The backup PRR with the requested function will not be put into use until the reconfiguration is finished. When the deployment of the partial bitstream on the backup PRR is finished (the requested function is ready), the port map of the interconnect system will be updated to forward the related data flows to the backup PRR, and will stop forwarding data to the previous active PRR. The previous backup PRR becomes active, and the previous active PRR assumes the role of the new backup PRR for future function reconfiguration.

As Figure 4.8 shows, the PRRs, interconnect system, and the interfaces should be properly orchestrated during the reconfiguration process. The central controller is responsible for orchestrating the reconfiguration process. The controller needs to perform five steps. Partial bitstreams of the requested function are downloaded from the host PC to the backup PRR in step (a). The traffic needs to be buffered in step (b). The controller then updates the port map of the interconnect in step (c). The traffic will then be released from the interface and will be forwarded through the PRR equipped with the new requested function in step (d).

The proposed reconfiguration process can improve the performance of the reconfiguration since the platform is still functioning during the downloading of the partial bitstream. The traffic only needs to be buffered when the port map of the interconnect is being updated. The buffer times to enable packet loss-free for both the common reconfiguration process and the proposed process are compared in the following paragraphs.

In the common process of partial reconfiguration, traffic needs to be buffered during the downloading of the bitstream to enable packet loss-free reconfigurable services since the PRR does not function during the reconfiguration period. Figure 4.9 shows the required download times of the partial bit files according to their sizes. The partial reconfiguration is requested through Vivado using the typical configuration of the reconfiguration controller (ICAPE2 at 100 MHz). As the partial bit file needs to be sent to the reconfiguration controller, the size of the bit file strongly influences the reconfiguration time. Table 4.2 shows the resource utilisation, size

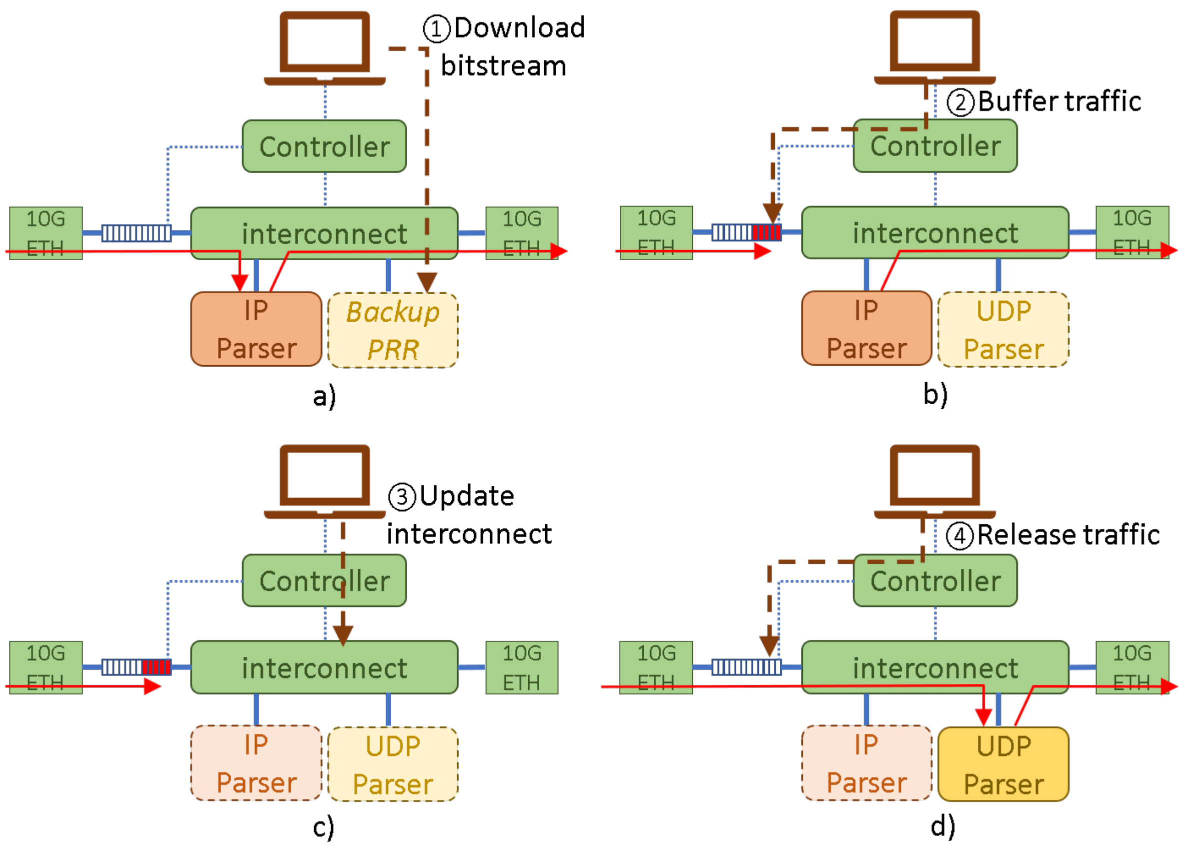

FIGURE 4.8. The proposed reconfiguration process that supports function reconfiguration in real time. It has the following four steps: a) Download partial bitstream to the backup PRR, b) Buffer the traffic, c) update the interconnect. d) Release the traffic

of partial bit file, and download time of a set of 64-bit network functions. The Ethernet parser, which has a minimum partial bit file size, needs 1,785 microseconds to be downloaded. The UDP parser takes the maximum amount of download time, which is 2,115 microseconds. Following the common reconfiguration process, these times translate to excessive buffering resources as Figure 4.9 shows.

During the proposed reconfiguration process, data flows only need to be buffered when the interconnect system is being updated since the previous function would still be running in the active PRR. The buffer time required in this process is only the update time of the interconnect system. In addition, the size of the partial bitstream will not affect the buffer time. As mentioned

Table 4.2: Resource Utilization and Partial Bitfile Size

VNF processor	Flip flop	LUT	Partial bit file size (KB)	Download time (us)
Ethernet Parser	493	290	714	1785
IP Parser	1268	1001	792	1980
UDP Parser	2072	1451	846	2115

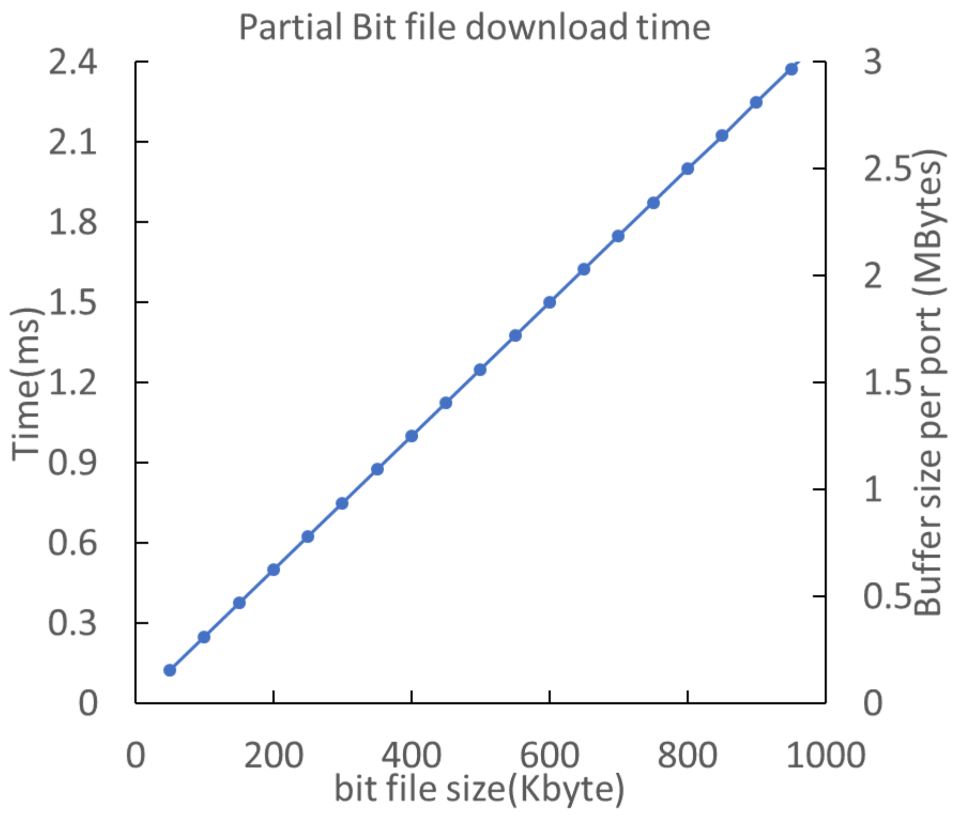

FIGURE 4.9. Time and buffer needed for packet loss free partial bitfile download.

in Section 4.3, the port map of interconnect system is updated through the AXI4-lite interface. The size of the update data will influence the buffer time. The size of each destination-output port pair of the forwarding table is 8 bytes. Figure 4.10 shows the time and buffer needed to reconfigure the interconnect system in packet loss-free manner.

The results indicate that the time needed to update an interconnect system is at the microsecond level (figure 4.10) compared to the common partial bitstream download time which takes anywhere from several hundreds of microseconds to several milliseconds (figure 4.9). As the proposed NFV platform is still able to process data flows during the partial bitstream's download

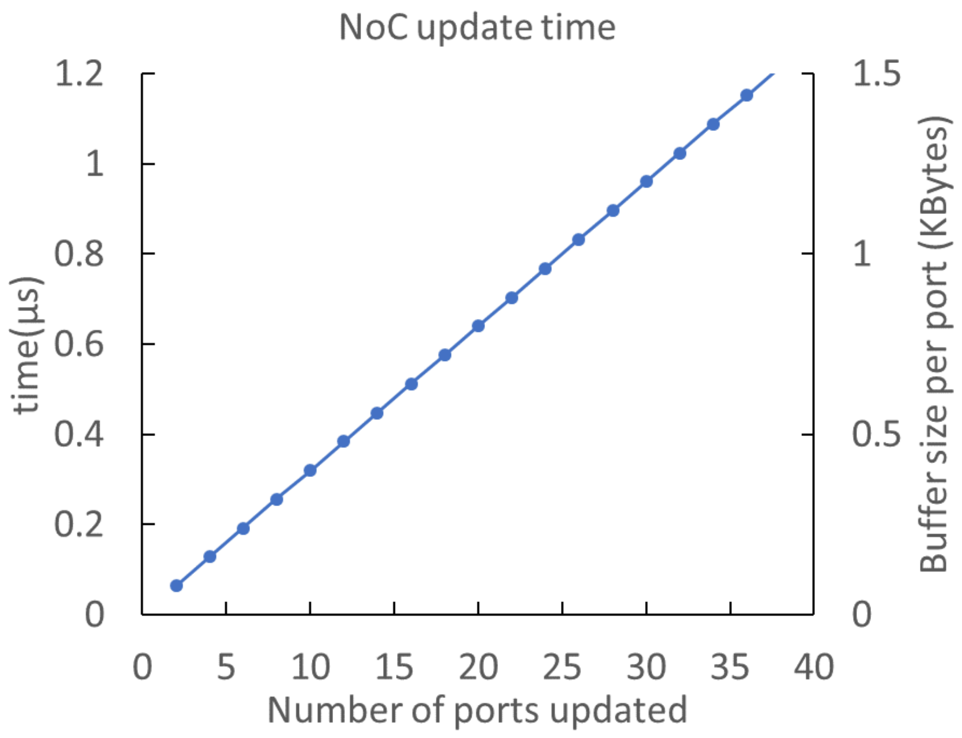

FIGURE 4.10. Time and buffer needed for packet loss-free NoC update time.

time, the proposed reconfiguration process can reduce the time required to buffer the traffic. Therefore, the proposed process enables a packet loss-free reconfiguration between network functions where no data or packets are lost during the process. By applying the proposed process, network developers will be able to change network functions on demand without stopping traffic.

4.6 Experiment and Demonstration

Table 4.3 shows the resource requirements for an interconnect system with 16 ports. The NFV platform is implemented on the NetFPGA SUME board with a Xilinx Virtex-7 690T FPGA chip [90]. The Network Master Pro MT1000A by Anritsu is used to generate and analyse the proposed NFV platform. The latency of the traffic analyser is accurate by 100 nanoseconds. The traffic analyser also has a built-in latency of 100 nanoseconds (measured in a loopback mode). This built-in latency has been removed in the following results. The network analyser is connected with the 10-Gbps SFP+ optical transceivers on the NetFPGA through a one-meter single-mode fibre. As the traffic analyser is accurate by 100 nanoseconds, the latency of the fibre (5 nanoseconds per metre) is ignored.

Table 4.3: Resource Utilization

Component	Flip flop	LUT	BRAM
NoC interface	446	267	714
NoC router	10398	22268	0
Central controller	994	432	2

An Ethernet switch is set up to demonstrate the interconnect system on real hardware. First, the interconnect system runs at the frequency of a 10G Ethernet system (156.25 MHz). Next, the interconnect system is run at a higher frequency (200 MHz). As Figure 4.11 (a) shows, in the implemented virtualisation system, the 10G Ethernet system at 156.2 MHz is directly connected to the interconnect system. Moreover, when the system is at 200 MHz (Figure 4.11 (c)), the clock converters are inserted between the 10G Ethernet system and the interconnect system. The amount of traffic on both configurations is the same: they receive two data flows and forward the data flows to the same 10G Ethernet output interface.

The influence of throughput on the latency is shown in Figure 4.11 (c). Additionally, the influence of packet size is shown in figure 4.11 (d). The 156.25-MHz system can achieve ultra-low latency figures (500 nanoseconds and 1 microsecond for minimum and maximum throughputs, respectively). For the 200-MHz system, the latency figures are 1.2 microseconds and 1.9 microseconds for minimum and maximum throughputs, respectively. The clock converter pf a 200-MHz system introduces more delays because the clock converter needs to insert invalid clock cycles (low valid clock cycles in AXI4-stream) in a packet when it is converted from a clock domain with a higher frequency (200 MHz) to a clock domain with a lower frequency (156.25 MHz). However, the transmitting side of a 10G Ethernet system always needs a complete packet (i.e. no invalid clock cycles in the middle of a packet) from the inner system of the FPGA. Thus, there must be a FIFO to (a) store a complete packet, (b) eliminate invalid clock cycles introduced by the clock converter, and (c) send the entire packet to the 10G Ethernet system. This store-forward FIFO increases the latency. Therefore, the NFV platform runs at a 10G Ethernet system's frequency, which is 156.25 MHz.

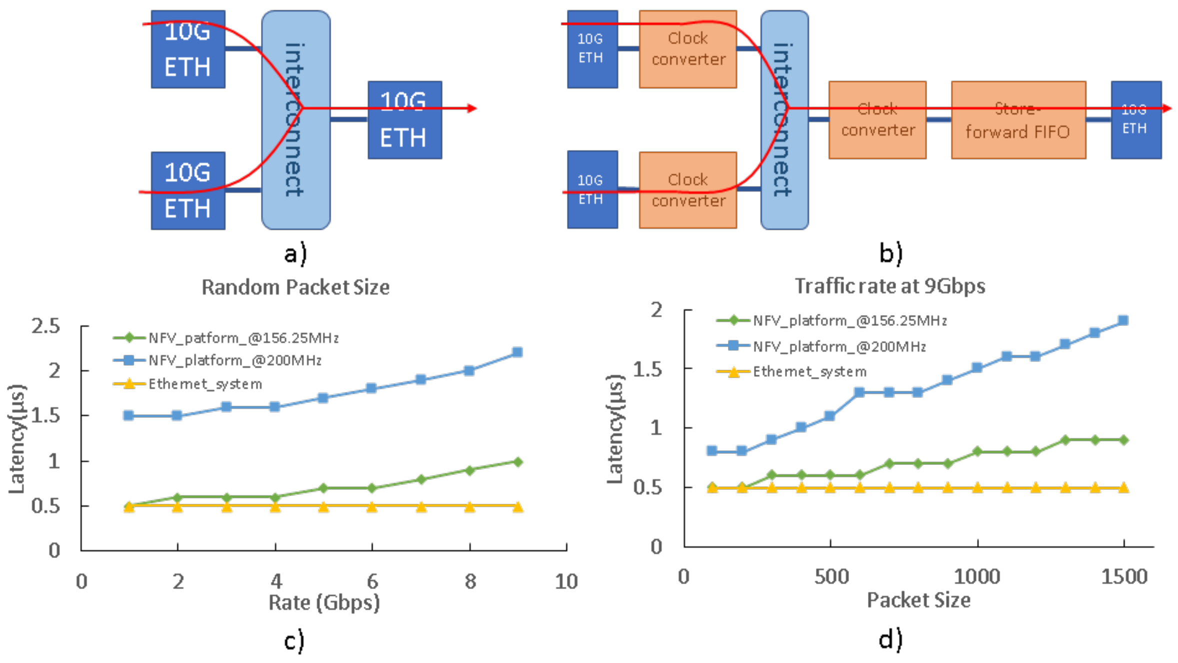

FIGURE 4.11. The experiment set up at 156.25MHz (a) and 200MHz (b). And the latency under different data rate (c) and Ethernet packet size (d).

Figure 4.12 demonstrates a reconfiguration between the IP and UDP parser. The IP parser is initially deployed in an active PRR and processes the current data flow. Then, the bitstream for the UDP parser is download in the backup PRR. After the proposed reconfiguration process, the traffic is forwarded to the backup PRR, so the deployed UDP parser starts processing the traffic. Next, the previous active PRR is configured into a UDP parser, and the traffic switches back to the active PRR by using the proposed reconfiguration process.

The number of parsed packets in each PRR is recorded as shown in Figure 4.13. The backup PRR parses the traffic from the 20th to 30th second. The packets are processed in the backup PRR from the 30th to 70th second. The previous active PRR is then configured into the UDP parser and is then put into use after 70 seconds. The backup PRR becomes available again for future reconfigurations. The traffic analyser records the total number of packets, which in this case is 77,518,546. The number of packets parsed in each PRR is 42,708,766 (for the original active PRR) and 34,809,780 (for the original backup PRR). No packets were lost during the whole process as the total number of counted packets in the FPGA is the same as the total number of

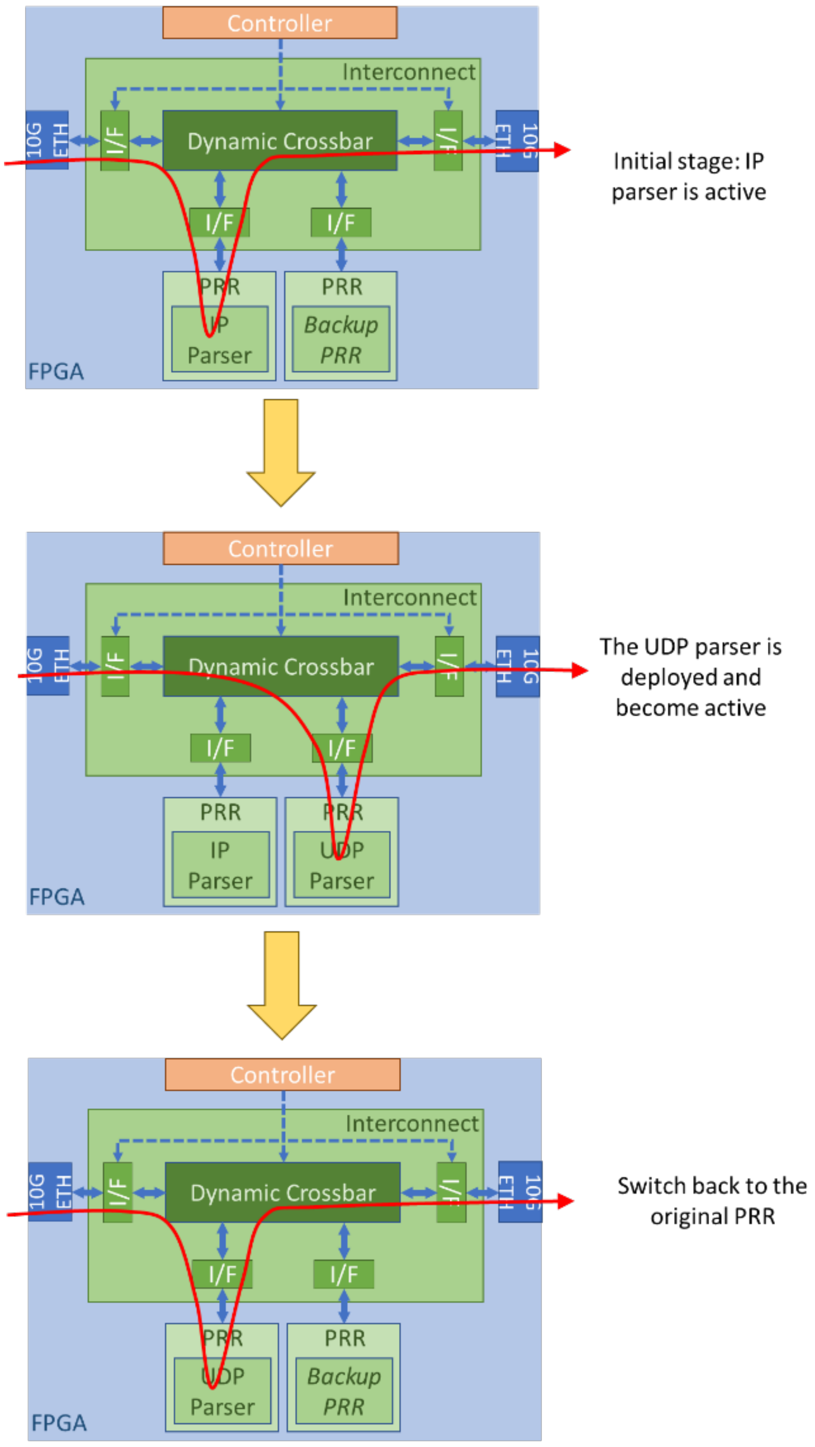

FIGURE 4.12. The demonstrated reconfiguration process.

transmit packets recorded by the traffic analyser (34,809,780 + 42,708,766 = 77,518,546).

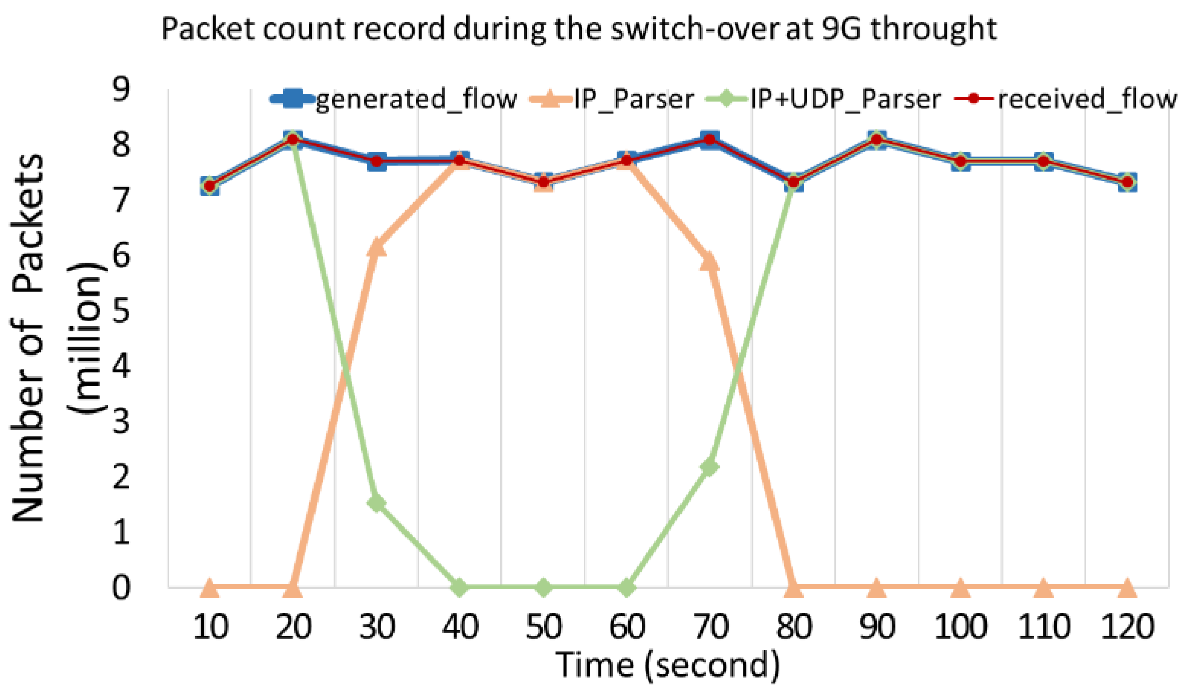

FIGURE 4.13. Network function virtualization in data centres.

4.7 Conclusion

This chapter has proposed an architecture that would isolate and share FPGA resources with multiple tenants of data centres and virtualise resources on FPGA platforms. As a example, the proposed architecture is applied to deliver flexible network stream processing. It is able to (a) keep both the bandwidth and the flexibility of network services, and (b) meet the requirements of recent programmable network technologies such as SDN and NFV.

An FPGA virtualisation architecture has been implemented. It includes several partial reconfigurable regions communicating through an on-chip interconnect system. The performance of the on-chip interconnect system is compared with the existing network-on-chip architecture. The result shows that the proposed interconnect can support higher bandwidths of up to 4.3 Gbps under random traffic at 156.25 MHz. A reconfiguration process to switch partial reconfigurable network functions in real-time has been proposed and demonstrated. In this process, network functions can keep running while the partial bitstream is downloaded. Experiments have been done to evaluate the proposed platform under real traffic. In the experiment, the platform forwards network streams from two input ports to one output port. It is possible to run at a maximum of 9 Gbps with a latency of 1 microsecond.

CHAPTER 5

SYNCHRONIZING THE RECONFIGURATION ON DISAGGREGATED FPGA RESOURCES

5.1 Introduction

An FPGA usually acts as a reconfigurable accelerator alongside processors due to its feature of parallel processing and its reconfiguration abilities. More FPGA resources are involved in high-performance computing thanks to recent technologies such as high-level synthesis and partial reconfiguration. The previous chapter discussed the virtualisation of resources on a single FPGA platform. This chapter explores the method that combines resources on distributed FPGA platforms. Recent research has tried to attach an FPGA directly to the data centre network and disaggregate FPGA resources in data centres. Frameworks have been proposed to manage the FPGA resources in traditional data centres [91]. However, most of these frameworks manage FPGA resources in a deploy-and-run manner. In other words, FPGA resources are not delivering any functions during the deployment of new functions. As the deployment introduces downtime, disaggregated FPGA resources cannot be reconfigured at very high frequencies.

This chapter explores methods that can reconfigure disaggregated FPGA resources at higher frequencies. As disaggregated FPGA resources are usually connected by data centre networks, network solutions are proposed to achieve high reconfiguration frequencies on disaggregated

FPGAs. The contributions of this chapter include:

- Two methods and network protocols for high-frequency reconfiguration are proposed. Reconfigure information is attached to every packet in the first method. The second method applies an independent packet to carry the reconfigure information.
- Protocol processing units for the suggested two methods are implemented. The two suggested methods will be compared.

This chapter also demonstrates a reconfiguration process for traffic at 10 Gbps in the experiment. The reconfiguration is captured by the Xilinx debug core and traffic analyser. The influence of the implemented protocol processor on the latency is measured. The protocol processor is able to support maximum total traffic of 9 Gbps (out of which 8.1 Gbps correspond to regular data flows, and 0.9 Gbps represent flows for reconfiguration).

The rest of the chapter is organised as follows. Section 5.2 introduces work related to the resource disaggregation and the management of the reconfiguration of FPGA resources in data centres. Section 5.3 motivates the high-frequency reconfiguration on disaggregated FPGA resources. Section 5.4 introduces the proposed solutions and protocols and the implementation of the associated protocol processor. Section 5.5 compares and evaluates the proposed solutions. Section 5.6 demonstrates the high-frequency reconfiguration on disaggregated FPGA resources. Section 5.7 concludes this chapter.

5.2 Background

Data centre resource disaggregation proposes a customisable data centre architecture. Ideally, in this architecture, each server mainboard contains only a unique type of computing resource (e.g. CPUs, memory, accelerators). When connected to a dynamic network, these servers along with their associated resources can form virtual hardware platforms in real-time to support cloud applications.

Because each type of resource can be scaled independently in resource disaggregated data centres, operators are able to manage data centre resources in a more flexible and granular manner. Specifically, the efficiency of resource utilization can be improved, as data centre operators are able to allocate each type of resource in real-time according to the resource requirements of the deployed applications. Also, the cost to upgrade a data centre's hardware resources can be reduced as each type of data centre resource can be upgraded independently [34].

The disaggregation of memory resources was investigated first because of its homogeneous features. For example, research by [92] implements a system-level software that enables pluggable memory resources which can be accessed through a PCIe interface. Based on remote direct memory access technology [27], Ethernet accessible memories are also implemented in FaRM [26].

In addition to memory disaggregation, researchers also investigate the disaggregation of other heterogeneous resources in data centres. Specifically, network resources play an important role in resource-disaggregated data centres as traffic that is traditionally inside the servers is now transferred inside the network [93]. The disaggregation of heterogeneous devices is often based on the virtualisation of the PCIe interface and interconnect [29]. For example, research by [94] sets up a prototype system to share the PCIe interface and its attached heterogeneous resource with many remote CPUs.

Because most of the heterogeneous devices are connected to their host servers through PCIe interfaces, virtualisation technologies based on PCIe interfaces and switches provide solutions to disaggregate existing devices in data centres. However, compared to the Ethernet, PCIe interfaces and switches have low scalability. For example, a PCIe switch needs 19 Watts [95] while an optical switch needs only 5 Watts [96]. As a result, the PCIe-based solutions only support disaggregation at a board or rack level instead of at the level of the entire data centre. Therefore, efforts have been made to attach heterogeneous resources to high-scalability interfaces instead of PCIe [61].

Reconfigurable hardware is also another important resource in a data centre [57]. Particularly, FPGA devices are the best option to address the problem of custom combinational logic in data centre environments [97]. FPGAs are often attached with processors as reconfigurable accelerators. The processor is responsible for the scheduling and management of local FPGA resources. For example, partially-reconfigurable functions are treated and managed as threads by the processor in research [98]. Research by [58] suggests a global solution for the management of FPGA resources in the range of entire data centres.

The processor is responsible for both the control and data communication between functions in aforementioned approaches. However, involving processors in the data path between FPGA functions reduces the bandwidth of these functions. To bypass the processors, research by [62] proposes disaggregated FPGA resources where FPGA can directly communicate with the network. In this approach, the data streams are able to run more smoothly in the data centre as FPGAs are directly attached to the network.

The FPGA plays an extraordinary role in the reconfigurable computing because of its partially-reconfigurable feature. However, most existing approaches to managing FPGA resources in data centres mentioned earlier follow a deploy-and-run manner. The FPGA does not function during the deployment of functions, which in turn introduces downtime to the reconfigurable computing services. To build more dynamic distributed and disaggregated FPGA resources, this chapter suggests and implements new methods.

5.3 Motivation

This section explains the problem of reconfiguring multiple FPGA devices in the network and briefly introduces the solutions.

Figure 5.1 illustrates the main difference between the reconfiguration on (a) a single chip and (b) across disaggregated FPGA resources through data centres. The reconfiguration process on a single chip is triggered locally by signals in the RTL design of the system. The timing of reconfiguration requests arriving at the FPGA resources can be counted accurately in clock cycles and is predicted by simulation tools. However, in the data centre's situation wherein multiple FPGA chips shall be reconfigured, the reconfiguration requests need to be encapsulated in network packets and are transferred through the data centre network. The latencies of these reconfiguration request packets are difficult to predict because they might be affected by the routing strategy and network congestion.

Figure 5.2 illustrates an example of where the unpredictable latency of the request for reconfiguration resulting in an unexpected state of failure. In the example, the platform includes

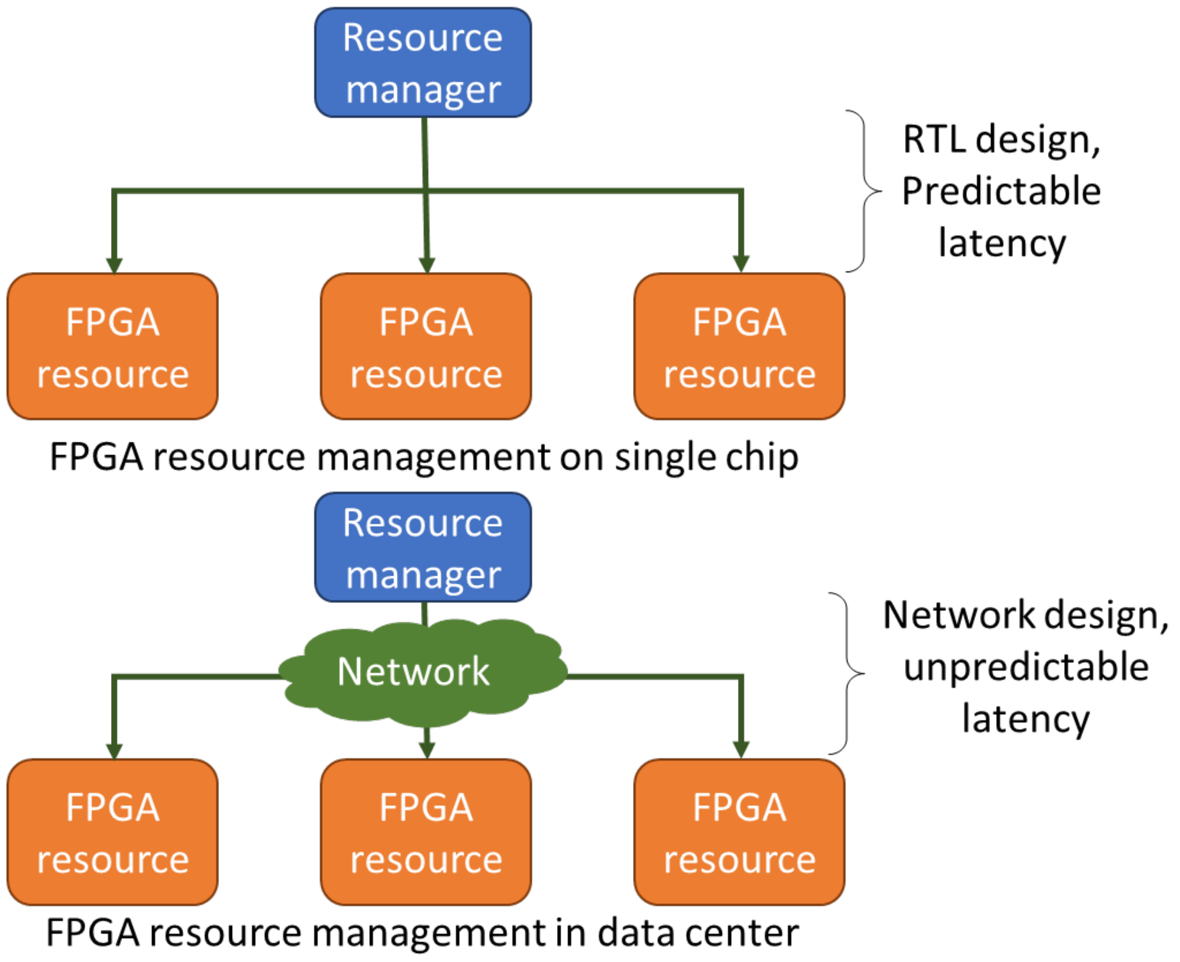
Resource manager
FPGA resource
FPGA resource
FPGA resource
RTL design, Predictable latency
FPGA resource management on single chip
Resource manager
Network
FPGA resource
FPGA resource
FPGA resource
Network design, unpredictable latency
FPGA resource management in data center

FIGURE 5.1. Reconfiguration of FPGA functions on single chip and in data centers.

two FPGAs that are attached to the network. First, the network data stream is encoded in FPGA1. The data stream is then forwarded through the network to FPGA2 and is decoded. The initial encoder1/decoder1 needs to switch to encoder2/decoder2 after the reconfiguration. Because of the unknown latency of the reconfiguration request, a state of failure (the encoder and decoder performed on the data flow do not match) may appear during the reconfiguration. Solutions in most approaches follow a deploy-and-run manner where the data stream is not sent until the resource manager receives all the reconfiguration-success responses from all the FPGA devices. This solution results in downtime during the reconfiguration because the data stream should be buffered during the distribution of requests.

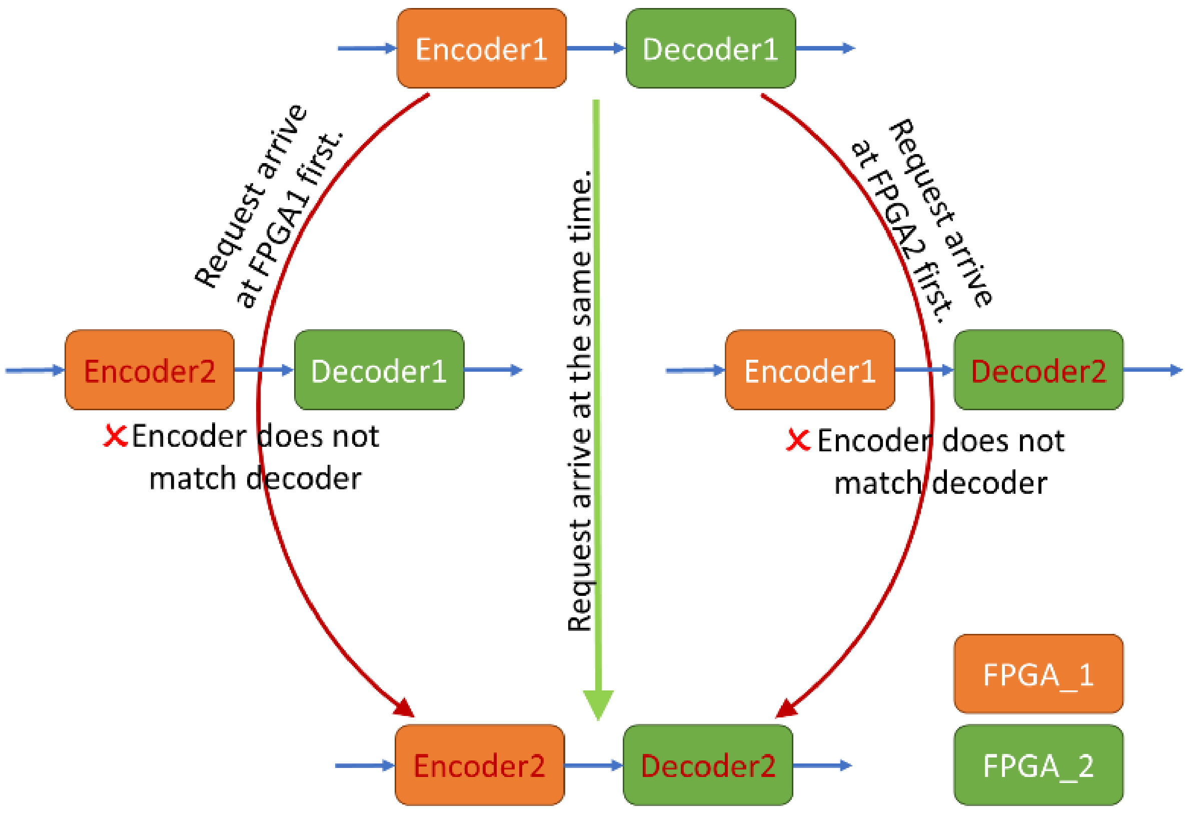
Encoder1
Decoder1
Request arrive at FPGA1 first.
Request arrive at the same time.
Request arrive at FPGA2 first.
Encoder2
Decoder1
✗Encoder does not match decoder
Encoder1
Decoder2
✗Encoder does not match decoder
FPGA_1
FPGA_2
Encoder2
Decoder2

FIGURE 5.2. An example for network-wide coherent function reconfiguration.

This chapter suggests two methods for reconfiguring the disaggregated FPGA resources without downtime or loss of services. Figure 5.3 shows the first method where the reconfiguration information is attached to every packet. Functions required by each packet are encapsulated as the header of the packet. When a packet arrives at an FPGA device, the FPGA device is able to reconfigure its functions according to the sequence of functions in the header. In this case, the data in a packet can only be processed by the sequence of functions that are indicated by its header. Therefore, the function reconfiguration can be synchronised to avoid unexpected states.

Figure 5.4 illustrates the second method where the reconfigure information is encapsulated in an independent packet. The source node/server inserts an independent packet in the data flow specifically for the function reconfiguration. The reconfigure information of all the FPGAs on the path of the data flow is encapsulated in that packet. The reconfigure packet will go through all the FPGAs along the path together with the data flow. When a reconfigure packet reaches an

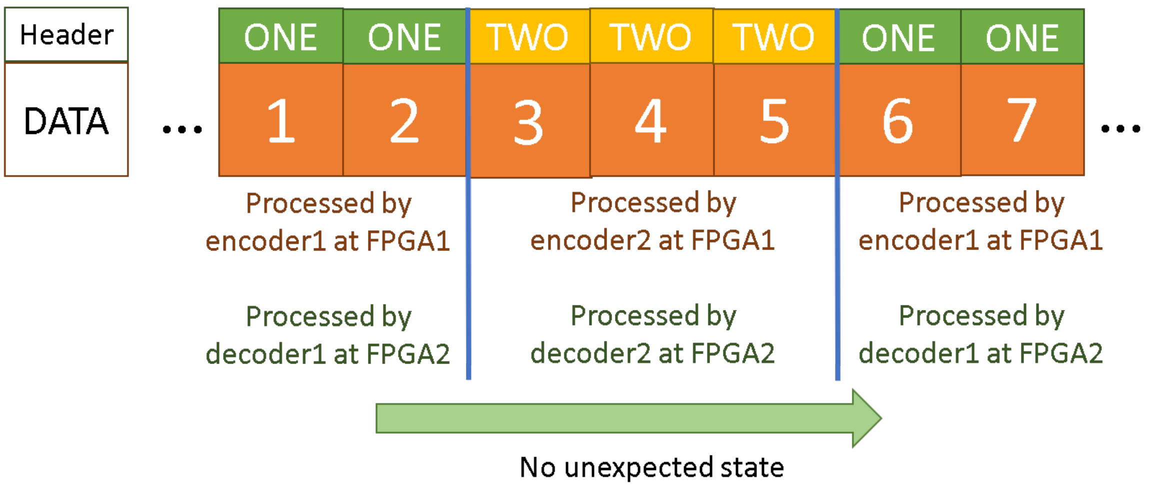
Header
DATA
...
ONE ONE TWO TWO TWO ONE ONE
1 2 3 4 5 6 7
...
Processed by encoder1 at FPGA1
Processed by decoder1 at FPGA2
Processed by encoder2 at FPGA1
Processed by decoder2 at FPGA2
Processed by encoder1 at FPGA1
Processed by decoder1 at FPGA2
No unexpected state

FIGURE 5.3. The first method: the reconfigure information is attached to every packet as headers.

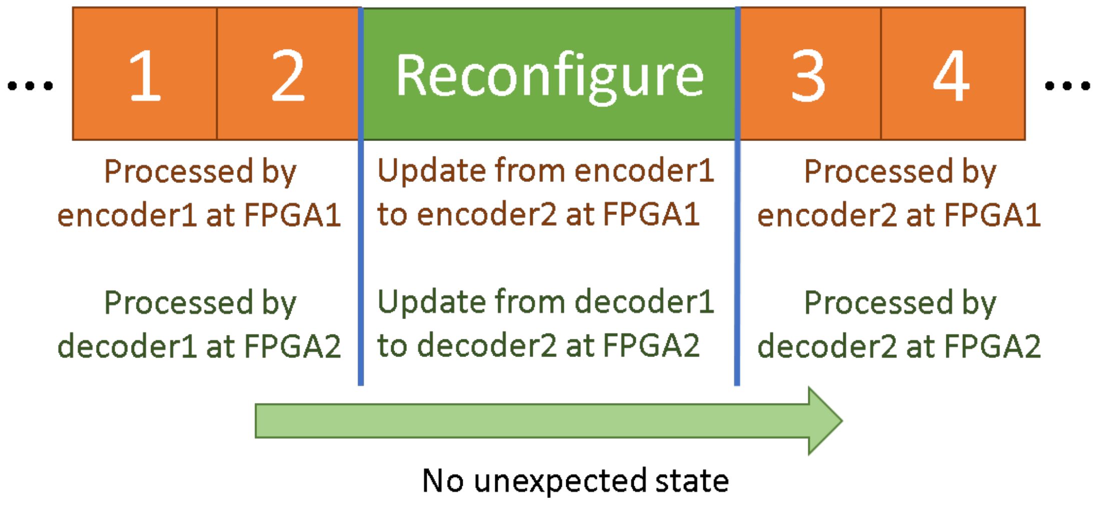
...
1 2
Reconfigure
3 4
...
Processed by encoder1 at FPGA1
Processed by decoder1 at FPGA2
Update from encoder1 to encoder2 at FPGA1
Update from decoder1 to decoder2 at FPGA2
Processed by encoder2 at FPGA1
Processed by decoder2 at FPGA2
No unexpected state

FIGURE 5.4. The second method: the reconfiguration is encapsulated in an independent packet.

FPGA, the data flow should be buffered. The reconfigure information carried by the packet for the current FPGA is extracted to reconfigure the function of the current FPGA accordingly. The data flow and the reconfigure packet are then released and are forward to the next FPGA. As Figure 5.4 shows, reconfigurable functions on different FPGAs are switched at the same location of the data flow (the location of the reconfigure packet). As a result, the inter-medium unexpected state can be avoided.

5.4 Implementation

The implementation of the two aforementioned methods is based on the reconfigurable platform and reconfiguration process reported by [85]. As Figure 5.5 shows, the reconfigurable platform includes several partial reconfigurable regions (PRR) and an interconnect system. The interconnect system is also connected with a 10-Gbps Ethernet system to communicate with off-chip networks. The platform also includes a controller to manage the interconnect system and orchestrate the reconfiguration process of the deployed functions.

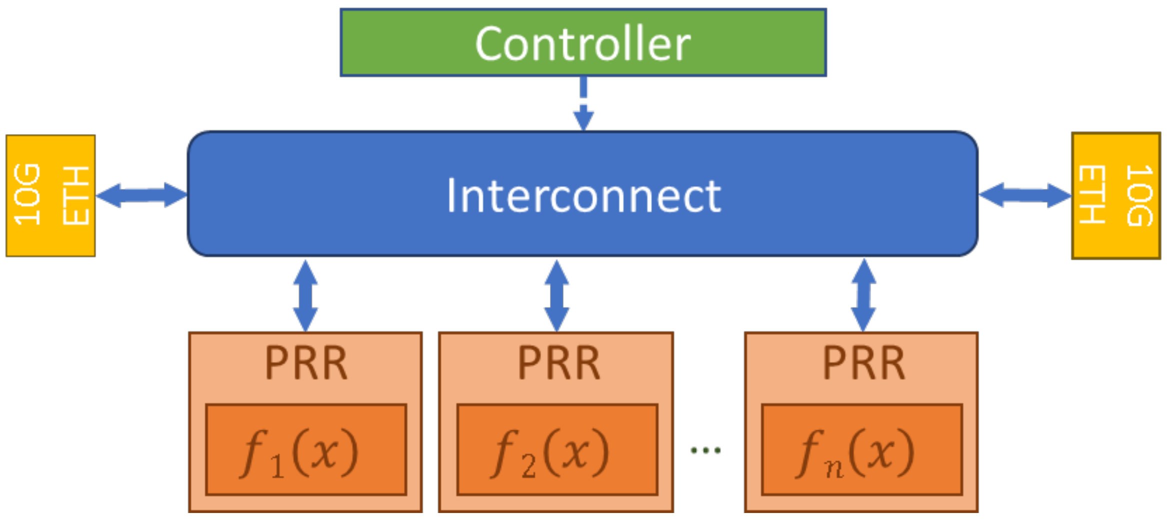

FIGURE 5.5. The overview architecture of the reconfigurable platform. Both of the proposed two methods are based on this platform.

As Figure 5.6 shows, instead of occupying all PRRs with reconfigurable functions, a backup (empty) PRR is reserved specifically for the reconfiguration process. Because network traffic has not been forwarded to the backup PRR, the partial reconfiguration on this backup PRR can be performed without disturbing existing traffic in the system. When a reconfiguration is requested (e.g. from IP parser to UDP parser), a partial bitstream of the requested function will first be configured on the backup PRR. The backup PRR with the deployed requested function will not be used until the partial reconfiguration is finished. When the deployment of a partial bitstream on the backup PRR is finished (the requested function is ready), the port map of the interconnect system is updated to (a) forward related data flows to the backup PRR with new requested function and (b) stop forwarding data to the previous active PRR. Once the reconfiguration process is complete, the previous backup PRR becomes active and the previous active PRR assumes the role of the new backup PRR for function reconfiguration in the future. As the reconfigurable platform is still able to process data flows during the download time of the partial bitstream, the proposed reconfiguration process can reduce the time required to buffer the traffic.

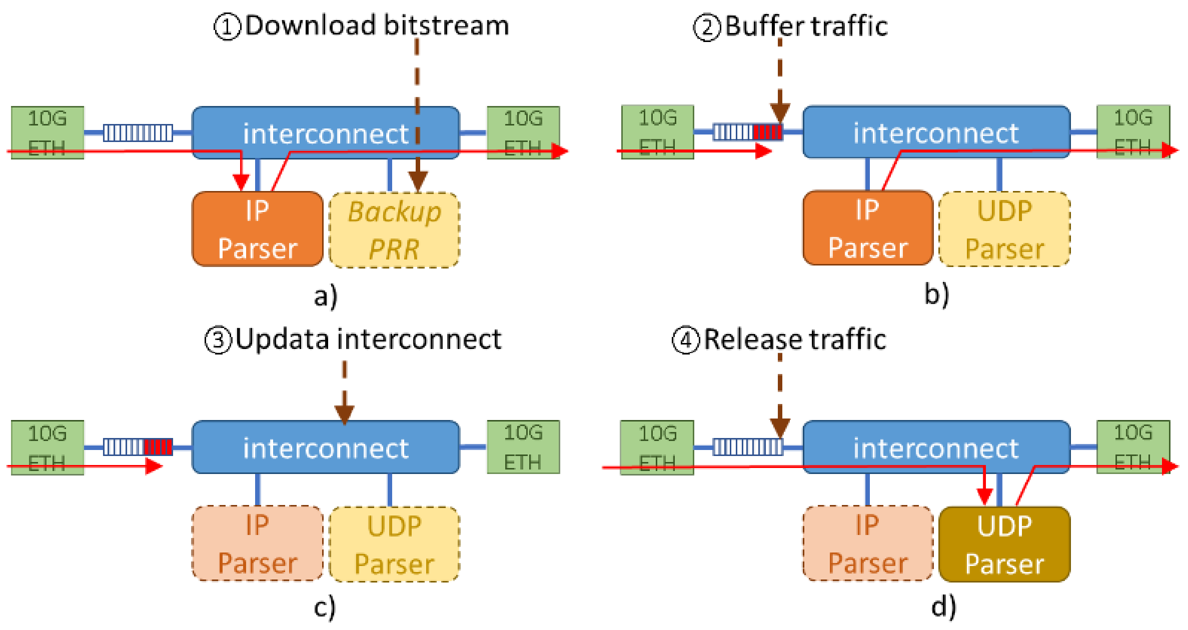

FIGURE 5.6. The reconfiguration process that supports real-time function reconfiguration. It includes four steps: a) Download partial bitstream to the backup PRR. b) Buffer the traffic. c) Update the interconnect. d) Release the traffic. It includes four steps: a) Download partial bitstream to the backup PRR. b) Buffer the traffic. c) Update the interconnect. d) Release the traffic.

As Figure 5.7 shows, a protocol processor is added between the 10G Ethernet and the interconnect system in the previous platform to achieve the two proposed methods. The detailed structure of the protocol processor is shown in Figure 5.7 left. The protocol processor is composed of the FIFO, FIFO controller, protocol processing unit, and master AXI4 interface. The received data flows are sent directly from the 10-Gbps Ethernet interface to the FIFO. When the FIFO controller detects a header that contains reconfigure information, it informs the FIFO to start buffering the traffic, and it also informs the protocol processing unit to begin extracting reconfigure information from the current packet. Then, the protocol processing unit extracts address and data information. The protocol processing unit then transfers the extracted data and address information to the master AXI4 interface to handle the handshake of memory-mapped AXI4 and updates the routing table of the interconnect system. When the AXI4 master module receives the successful responses of all the transfers, the protocol processing unit informs the FIFO controller to release the traffic.

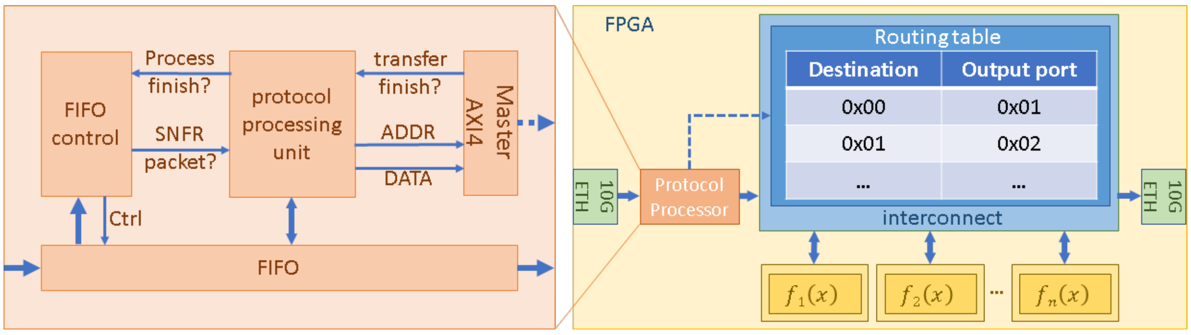

FIGURE 5.7. The location and architecture of the protocol processor.

Figure 5.8 illustrates the data structure of the protocol to support the proposed two methods. In the packet to support the method of attaching reconfigure information to every packet (Figure 5.8 (a)), the functions requested by the current packet are encapsulated in this packet as additional reconfigure information. Specifically, each packet includes addresses and data used to update the port map of the interconnect system and change the functions performing on the current packet as reconfigure information. Figure 5.8 (b) illustrates the data structure for the packet to support the method of independent packets. The reconfigure information is encapsulated as the payload of IP protocol in an independent packet. This packet which is marked by the protocol field in IP protocol is only responsible for the reconfiguration of functions and does not carry any real data in the flow. The structures of the reconfigure information are the same in both methods; only the location of the reconfigure information differs. The definitions of each field of the proposed protocol are listed as follows:

- Offset: 64 bits. This field marks the starting point for 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 address. If it is 0xFFFF_FFFF, the protocol processing unit should stop extracting data and the address.
- DATA: 32 bits. The data that should be transferred to the memory-mapped AXI interface.

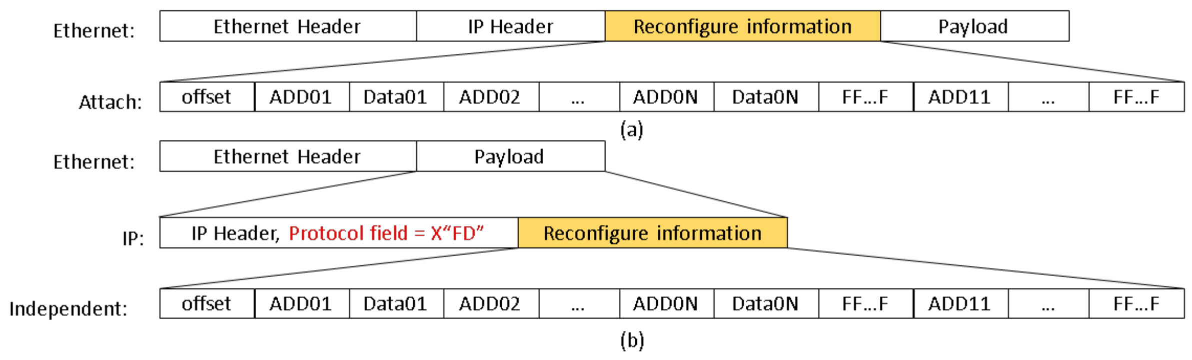

FIGURE 5.8. The protocol for both method. (a) The protocol for the first method. The reconfigure information is in the header of every packet. (b) The second method. The reconfigure information is encapsulated in an independent packet. Thas packet is marked by the protocol field of the IP protocol.

Figure 5.9 illustrates the behaviour of the protocol processing unit. Because the structures of the reconfigure information are the same in both methods, the behaviours of the processing unit are also the same. When a packet is received, the processing unit will extract the address and data information from the location pointed by the offset field. A specific address value 0xFFFF_FFFF marks the finish location of the extraction process. When the address and data have been extracted, the processing unit will update the offset value which points to the location of next pair of address and data. The packet will then be sent to the next FPGA in the path. The next FPGA will read the updated offset, and the information for the previous FPGA can be skipped. It should be highlighted that the reconfigure information in each packet carries the update information (address and data) for all the associated FPGAs in the path. The address field with a value of 0xFFFF_FFFF divides the reconfigure information into many segments that reconfigure different FPGA devices.

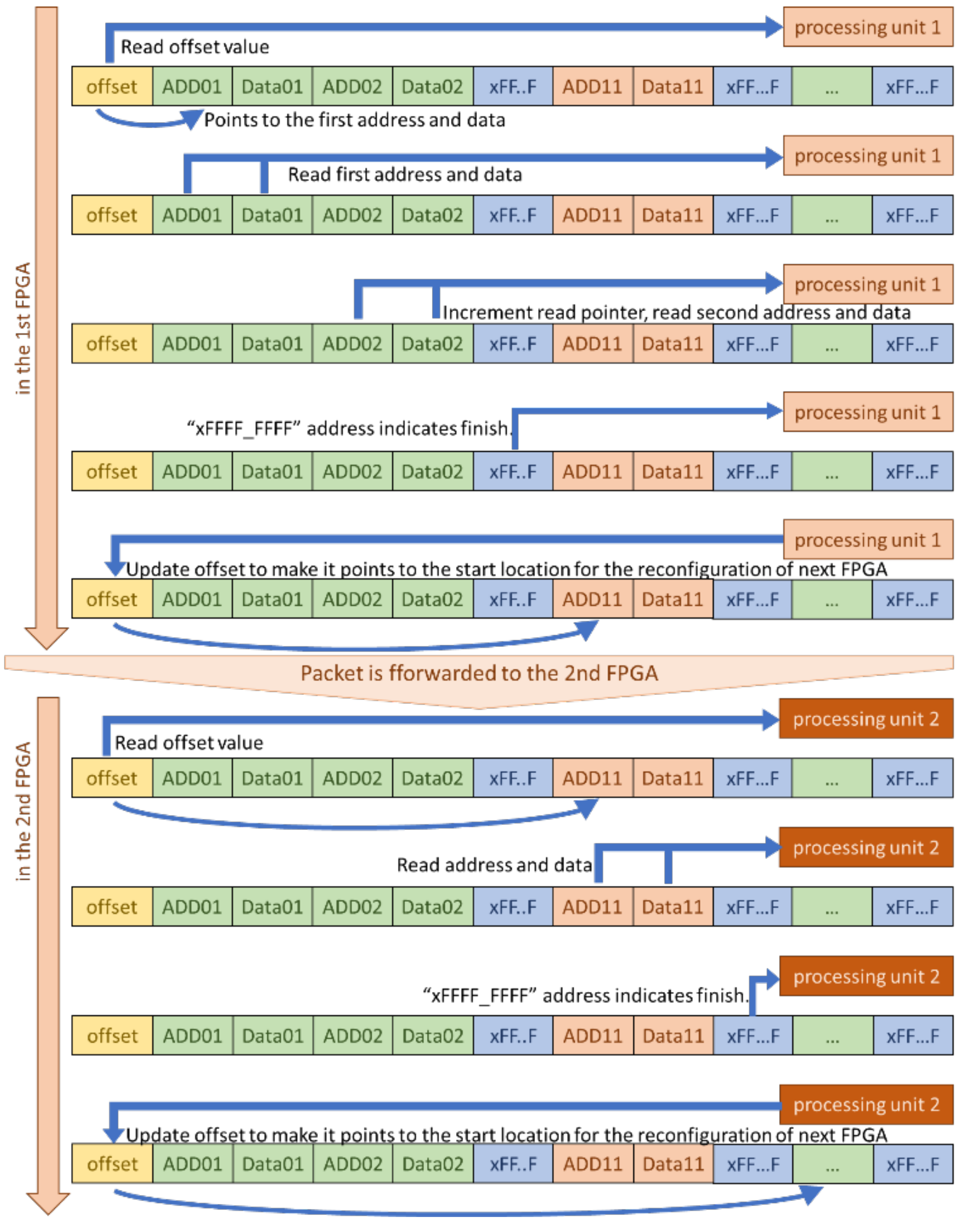
in the 1st FPGA
processing unit 1
Read offset value
offset
ADD01
Data01
ADD02
Data02
xFF..F
ADD11
Data11
xFF...F
...
xFF...F
Points to the first address and data
Read first address and data
Increment read pointer, read second address and data
"xFFFF_FFFF" address indicates finish
Update offset to make it points to the start location for the reconfiguration of next FPGA
Packet is fforwarded to the 2nd FPGA
in the 2nd FPGA
processing unit 2
Read address and data

FIGURE 5.9. The brief description of the behaviour 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 to the start location for the next FPGA.

Figure 5.10 illustrates the detailed state machine of the protocol processing unit. When a packet with reconfigure information is received, the traffic will be buffered. The processing unit first checks the offset of the received packet. Then, the processing unit sets the read address of the BRAM in the FIFO to the offset value to read the first address and data pair. The value of the extracted address is checked: if it is not 0xFFFF_FFFF, then the processing unit will increment the read address to read the next address and data pair. If the address is 0xFFFF_FFFF, the processing unit will then update the offset field of the received packet to make it point to the address of and data for the reconfiguration of next FPGA. The received packet and the traffic will be released in the next state. Thus, the received packet with the updated offset can be sent to the next FPGA in the path.

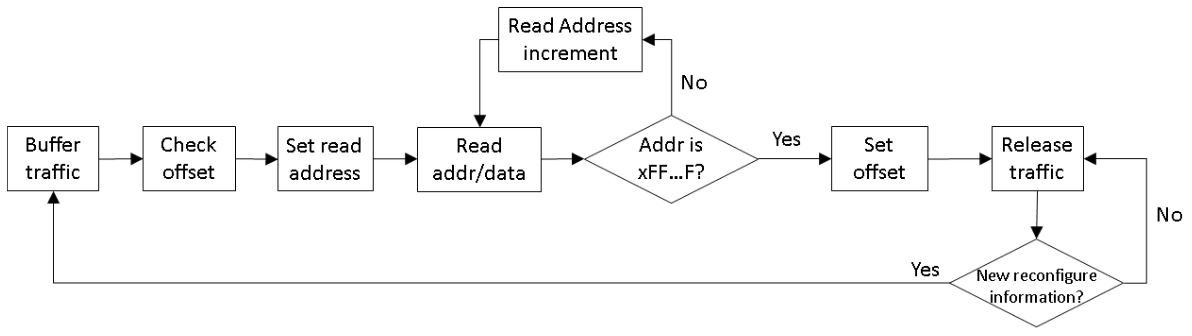

FIGURE 5.10. The state machine of the protocol processing unit.

5.5 Evaluation

This section evaluates the suggested two methods for the function reconfiguration over disaggregated multiple FPGAs. A clock cycle-accurate simulation has been established to measure the latency and maximum data rate of the two suggested methods. All the modules run at the frequency of the 64-bit Ethernet system: 156.25 MHz. The simulation set-up is shown in Figure 5.11. The function of the protocol processor is modified according to the method under evaluation. The traffic generator produces packets, which analysers consume. Traffic attributes such as packet size and data rate can be modified in the traffic generator. The analysers capture all the packets, and the latency is calculated at the analysers. The interconnect system has a built-in latency of two clock cycles, which has been eliminated in the results.

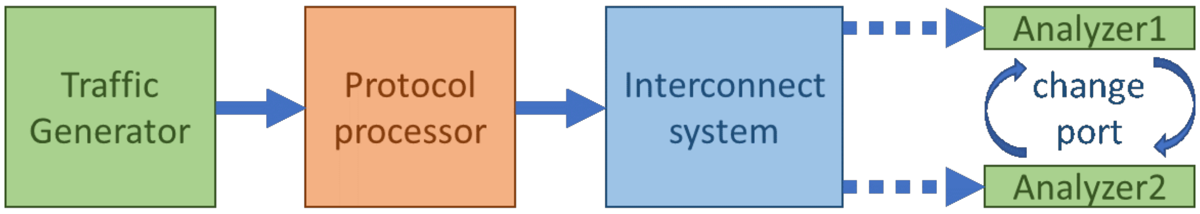

FIGURE 5.11. The simulation set up to compare the two proposed methods.

The traffic generator is responsible for inserting reconfigure information into the regular traffic flows. According to the method under evaluation, the traffic generator provides a different format for reconfigure information. In evaluating the first method, the reconfigure information is attached to every packet and packets are generated with the data structure as Figure 5.8(a) shows. In addition, when the second method is being tested, the traffic generator sends independent packets for reconfiguration with the data structure as Figure 5.8 (b) illustrates. The reconfiguration performed in evaluating both methods periodically changes the output port of the interconnect system to indicate that regular traffic can flow smoothly in the reconfigurable system without distinct downtime.

Figure 5.12 shows the results of the first method where the reconfigure information is attached to every packet. As the reconfigure information is attached to each packet, the output port of the interconnect system is reconfigured every packet. The load-latency curve is collected. As Figure 5.12 (a) illustrates, the packet size has an impact on the latency. A smaller packet size means that more headers need to be processed in the protocol processor and increases in the latency accordingly. Figure 5.12 (b) illustrates the relationship between the maximum data rate and packet size. Because the traffic is buffered when the protocol processor extracts information from the header of a packet, more headers increase the traffic buffer time and reduce the maximum rate.

The packet size heavily influences the performance of the first method (attaching reconfigure information to every packet). When the packet size is smaller than 600 bytes, the performance of the protocol processor is severely reduced in terms of both latency and maximum data rate. However, when the packet size of traffic is small, reconfigurable computing can be performed over

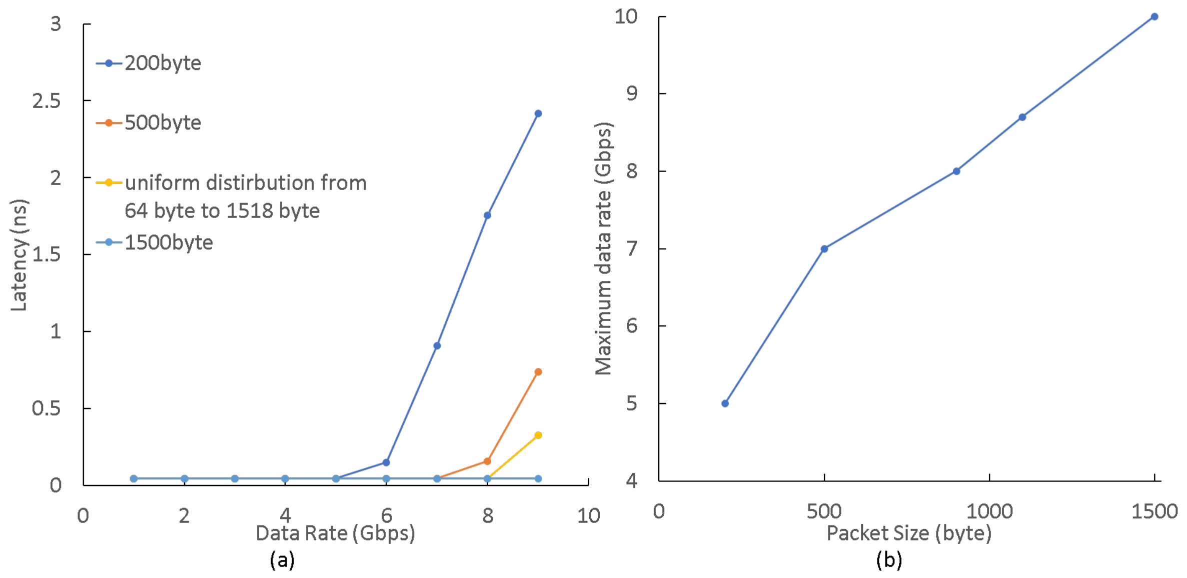

FIGURE 5.12. Simulation results for the first method. (a) The influence of packet size on the latency. (b) The maximum data rate for different size of packets.

more fragile data. Functions that exist on disaggregated FPGA resources can also be reconfigured at higher frequencies when the packet size is small.

Figure 5.13 shows the results of the second method where the reconfigure information is sent using an independent packet. The reconfiguration frequency is changed by configuring the frequency that sends the packet for reconfiguration to investigate the influence of the packet that carries the reconfigure information. Figure 5.13 (a) illustrates the relationship between the reconfiguration frequency and the latency. The latency increases sharply at a specific frequency. That is because when the frequency of the reconfiguration is high, the system does not have enough time to release all the buffered data between two reconfigure packets, the buffered data start to accumulate, and the latency increases significantly. Figure 5.13 (b) shows the relationship between the maximum frequency of reconfiguration and the data rate of regular data flows. The maximum frequency needed to reconfigure the system drops according to the increment of the data rate of regular flows. However, the system is still able to support 400 KHz at a data rate of 9 Gbps.

The second method is still able to reconfigure disaggregated FPGA resources at a reasonable

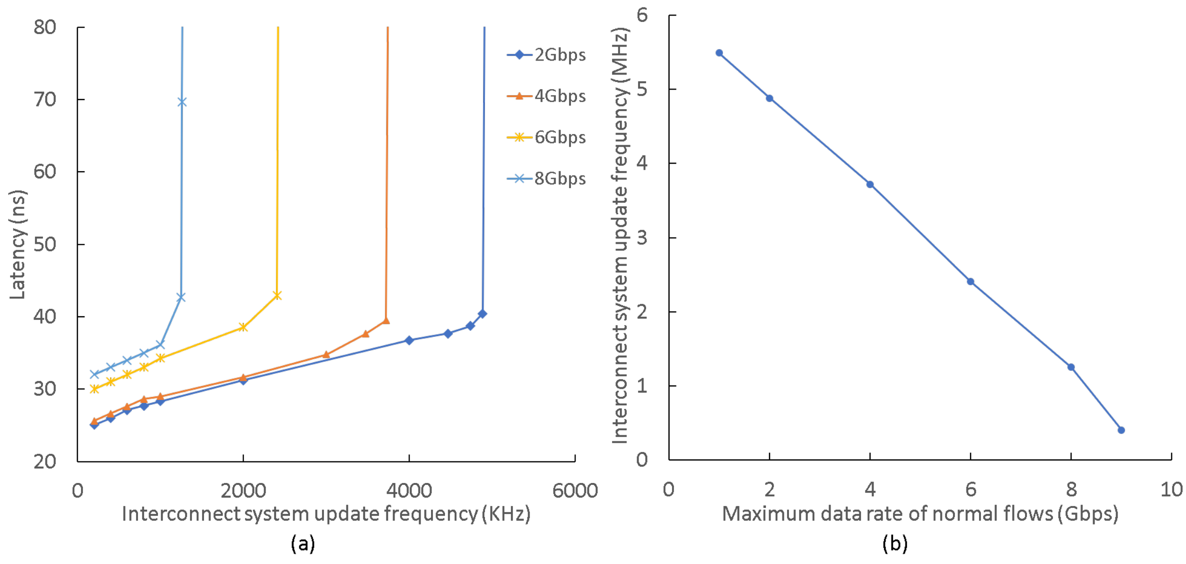

FIGURE 5.13. The simulation result for the second method. (a) The update frequency of the interconnect system has a threshold. (b) The maximum update frequency at different data rate.

frequency at the level of mega times per second. However, the performance can also be kept at a reasonable level. The latency of the system almost increases with the frequency until the reconfiguration frequency reaches its maximum supported value. Moreover, the system is still able to support a data rate of 9 Gbps if the system is not reconfigured at a very high frequency. However, as the reconfigure packet's location determines when the reconfiguration happens, this method only supports networks that strictly transfer packets in order.

Both methods are able to perform function reconfiguration over disaggregated FPGA resources without experiencing distinct downtime. The reconfiguration in the first method (attaching reconfigure information to every packet) is highly coupled with regular packets. This feature might cause problems in the management of data centre resources. As the resource controller is often a centralised system, it is difficult to determine if there is a packet that reconfigures FPGA resources or not because every packet has the option to reconfigure the FPGA resources. However, the reconfiguration in the second method takes place independently of regular packets. The central data centre resource controller only needs to keep track of the packet for reconfiguration to know the status of FPGA resources. However, the granularity of reconfiguration over data is low in the second method. However, as Figure 5.13 shows, there is a chance to trade off the reconfiguration frequency for the data rate. Another limitation of the second method is that it only supports the transfer of the data in order, but this is common in the recent resource disaggregated data centres where servers are optically and directly linked without any routing or other elements that disorder packets [99]. Therefore, the second method is chosen for the final demonstration of the reconfiguration of disaggregated FPGA resources.

5.6 Experiment

This section describes the experiment conducted to demonstrate reconfiguration on disaggregated FPGA resources without distinct downtime. The Xilinx debug core is utilised, and the traffic analyser is used to capture the reconfiguration of the network. The influence of reconfiguration 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 [100]. Table 5.1 illustrates the resource utilisation of the protocol processor.

Table 5.1: The resource utilisation 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

Figure 5.14 illustrates the experimental set-up and the reconfiguration process. Two FPGAs are set up and are connected through 10-Gbps Ethernet interfaces. A system is set up on each FPGA with an on-chip interconnect system, two reconfigurable regions, and a protocol processor. In each reconfigurable region, there is a packet counter to record the number of packets forwarded to each region. The traffic is generated and analysed in the traffic analyser. A 9-Gbps traffic with a random packet size is set up to represent the regular data flow in the data centre network. One reconfiguration packet is sent from the traffic analyser every 5 seconds. The generated

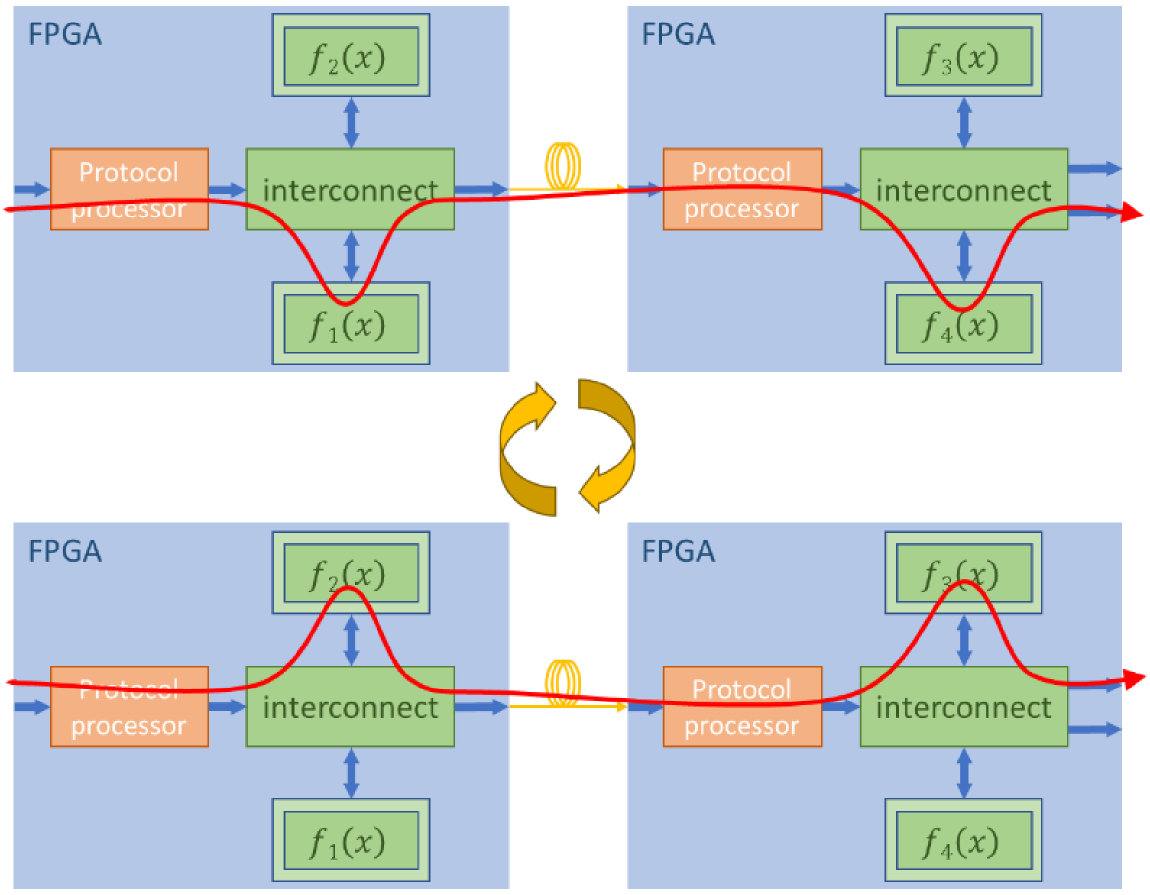

FIGURE 5.14. The demonstrated reconfiguration process.

reconfiguration packets cause the data flow to continuously and periodically switch between the following two states:

$f_1(x)$ at FPGA$_1$, $f_3(x)$ at FPGA$_2$, FPGA$_2$ forward traffic to output$_1$.

$f_2(x)$ at FPGA$_1$, $f_4(x)$ at FPGA$_2$, FPGA$_2$ forward traffic to output$_2$.

The state machine of the processing unit and the valid signals of both regions are monitored in the Xilinx debug core as shown in Figure 5.15 (a). When a reconfiguration packet is received, the reconfiguration process is triggered. The reconfigure packet is only processed when the traffic is completely buffered to ensure no loss of data during the reconfiguration. As Figure 5.15 (b) illustrates, the traffic analyser is also used to monitor the output of FPGA 2. The traffic is detected at Output 1 in the initial state. After reconfiguration, the traffic from Output 1 cannot be detected. Instead, the traffic from Output 2 is what gets detected.

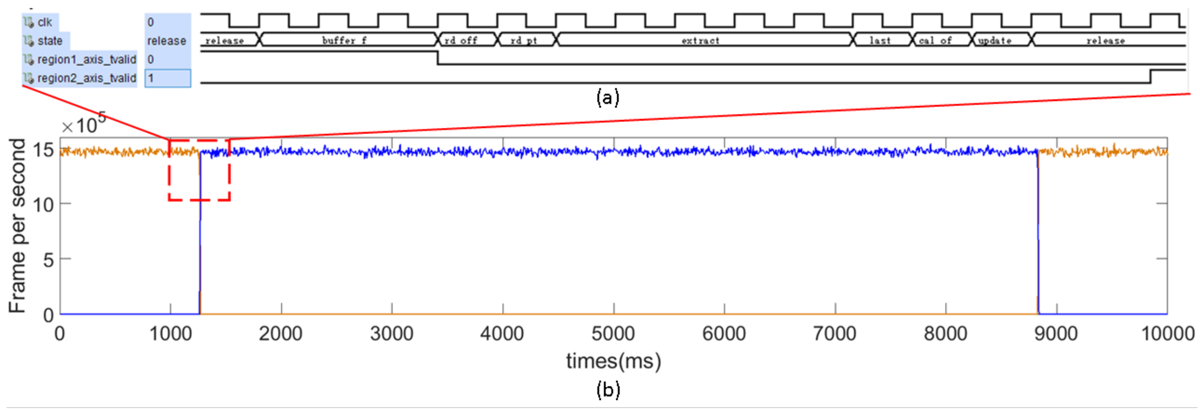

FIGURE 5.15. (a) The state machine and valid signal captured by the Xilinx debug core. (b) Reconfiguration of output port monitored by the external traffic analyser.

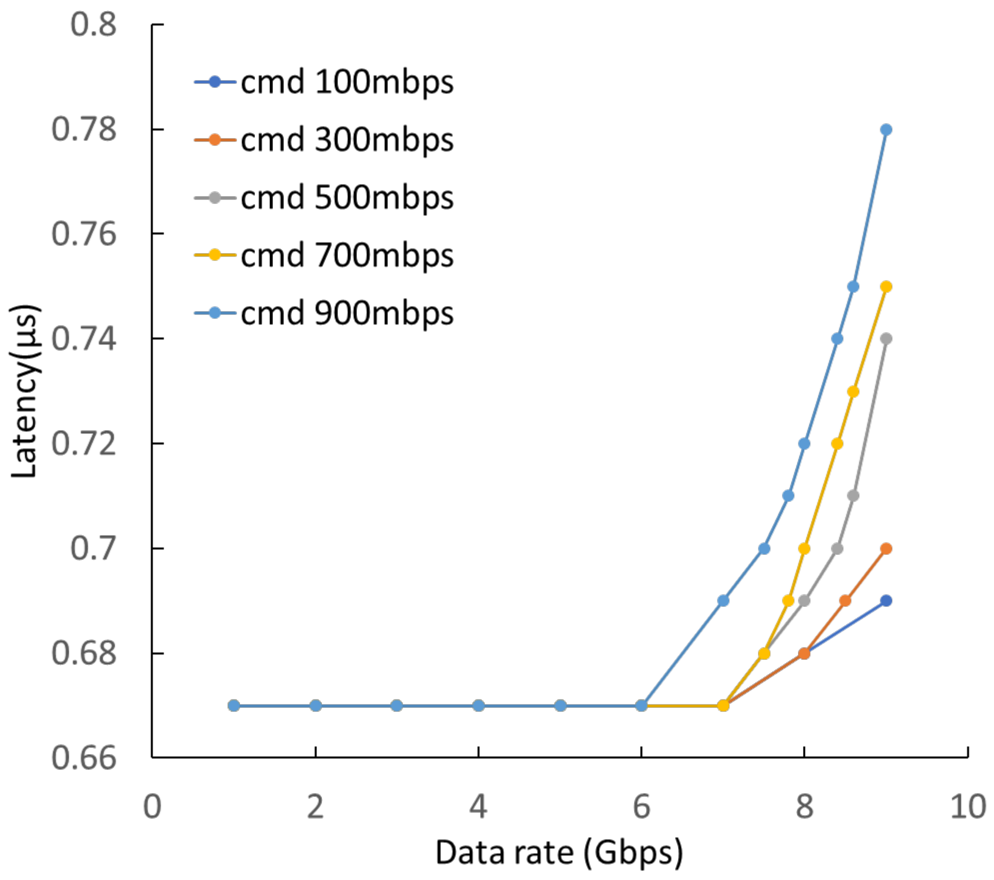

FIGURE 5.16. The latency influence of the protocol processing.

As Figure 5.16 shows, the influence of the reconfigure packet on the latency of the regular data flows is also measured. To reconfigure disaggregated FPGA resources, reconfigure packets should be inserted into the regular data flows. In this experiment, the reconfigure packets are inserted with different rates, and the data rate vs latency curve is measured to analyse the influence of the reconfigure packets. It should be pointed out that the reconfigure packet is responsible for changing the state of the network. Therefore, a higher reconfigure packet rate means the network reconfiguration takes places at a higher frequency. As Figure 5.16 illustrates, the reconfigure packet flows starts influencing the traffic when the data rate of regular data flows is higher than 6 Gbps. The data rate of reconfigure packets can reach 0.9 Gbps. Hence, it is able to reconfigure the FPGA resources at a high frequency without distinct downtime. The protocol processor also supports high throughput. It has the ability to work at a data rate of 9 Gbps (8.1 Gbps for regular data flows plus 0.9 Gbps for reconfigure packet flows).

5.7 Conclusion

This chapter has explored solutions that would enable high-frequency reconfiguration on disaggregated FPGA resources in data centres. This chapter has proposed two network solutions to propagate reconfigure requests. The reconfigure information is attached to every packet in the first solution. In the second solution, the reconfigure information is encapsulated in an independent packet. Protocol processors are implemented for the two solutions. The first solution, i.e. attaching reconfigure information to every packet, can perform reconfigurable computing over more fragile data and support disordered flows. However, the network performance, including latency and bandwidth, will be affected when the packet size is small. The performance of the second solution, i.e. encapsulating reconfigure information in a dependent packet, is independent of the packet size. It is also possible to trade off between the reconfiguration frequency and the data rate in the second solution. This chapter also sets up an experiment to demonstrate high-frequency reconfiguration. The Xilinx debug core and external traffic analyser capture the reconfiguration during the experiment. The protocol processor can support a high data rate Ethernet traffic of bandwidth up to 9 Gbps (8.1 Gbps for regular flows and 0.9 Gbps for reconfiguration flows).

Chapter 6

Conclusion

With the scale-up of data centres, the management of computing resources in a data centre becomes important. Therefore, this thesis has explored the management of data centre resources. The management of a data centre aims to increase the efficiency of resource utilisation. In other words, the management should help to meet more cloud application requirements at lower costs. The data centre resources should be easy to partition. Thus, they can be shared with many cloud services. Additionally, with the growing size of a data centre, it should be possible to combine large amounts of resources to support extremely high-volume computing.

The second chapter has reviewed methods that help to manage data centre resources. Virtualisation aims to partition the resources on a single platform into multiple pieces so that operators can conveniently allocate them to cloud applications. Virtual machine migration has been suggested to optimise the resources in real-time. During the migration, technologies are proposed to synchronise memory data on multiple servers. These technologies hint at the disaggregation of data centre resources as it is possible to combine physically distributed resources for executing single tasks. Specifically, the virtualisation and disaggregation of reconfigurable FPGA resources have been reviewed. High-bandwidth parallel stream processing is one of the features of the FPGA platform that stands out most. Therefore, this thesis has reviewed architectures that try to maintain this feature when introducing FPGA resources into data centres.

When software support for the disaggregation of memory resources has been developed, it has been found that the network highly limits the performance of the final computing systems as traffic that was previously on the board is now flowing in the data centre network. Therefore, developing a high-performance network has become a key enabler of resource disaggregation. In Chapter 3, a network interface for data centre servers has been implemented. In the network interface, optical transceivers equipped with advanced silicon photonic technology has been set up and combined with an FPGA board. Its bit error rate is compared with the bit error rate of a traditional SFP+ transceiver. The results show that the new transceiver is able to deliver high bandwidth and high-quality signals at lower footprint cost.

The reconfigurable devices such as FPGAs have become increasingly important in cloud computing as they are able to deliver flexible high-volume stream processing in parallel. Although the management of memory resource has widely been adopted in data centres, the management of reconfigurable FPGA resources has just begun. Therefore, in Chapters 4 and 5, architectures have been implemented to enable the virtualisation and disaggregation of FPGA resources.

An architecture that virtualises the FPGA has been implemented in Chapter 4. Resources on single FPGAs are divided into several partial reconfigurable regions. An interconnect system is established to enable communication between reconfigurable regions and I/Os. In this architecture, FPGA platforms are partitioned into several reconfigurable regions that multiple cloud applications can remotely access. As an example, the implemented virtualised FPGA platform is adopted as a platform for network function virtualisation. It can support a maximum of 9 Gbps traffic with the latency of 1 microsecond. The deployed network function can seamlessly be reconfigured in real-time.

Chapter 5 has explored methods that try to efficiently combine physically-distributed FPGA resources. Instead of sending requests from a central controller, network solutions have been proposed to trigger the function reconfiguration on FPGA devices. By inserting requesting packets between normal packets, the reconfiguration of multiple pipelined functions can be synchronised. The results have shown that the proposed method reduces the downtime during the function reconfiguration, and hence enables reconfiguration at high frequencies. The implemented processor for the protocol of requesting packets can support Ethernet traffic of up to 9 Gbps, 0.9

Gbps of which are packets for reconfiguration.

6.1 Achievement

The achievement of this thesis can be summarised as follows:

- In Chapter 3, a high bandwidth network interface has been set up by combining an OptoPHY optical transceiver with FPGA devices.
  - Experiments are set up to compare the OptoPHY transceiver with the SFP+ transceiver.
  - An optical switch is also set up to emulate the optical network.
- In Chapter 4, an architecture has been developed and implemented to virtualise the FPGA resources. The system is able to maintain the high-volume stream processing of FPGA platforms.
  - An interconnect system has been set up and it can switch the traffic based on Ethernet address.
  - A reconfiguration process has been demonstrated to enable low-down-time function switch-over.
- Chapter 5 has proposed a solution to synchronise function reconfiguration over multiple FPGA devices.
  - A network protocol has been designed and the protocol processor has been implemented.
  - The reconfiguration of multiple FPGA devices can be performed at high frequencies.

My research make the initial effort to introduce reconfigurable FPGA resource in the data centre. I explores techniques to set up the physical platform that attach FPGA directly to a pure optical data center network. The basic architecture of the platform has been established. When new optical transceivers are invented, the related modules in the platform can be upgraded accordingly. Architecture that isolates and shares FPGA resource on a single device are proposed.

By optimizing this architecture, I believe researchers are able to establish a virtual FPGA environment where users of the same FPGA device cannot feel the existence of other users. The reconfiguration on multiple FPGA devices is also proposed. This method helps researchers to do reconfigurable computing within the range of the data center instead of a single machine.

6.2 Lessons learned

The most valuable lesson I learned during PHD is how to do researches. Firstly, I should know the basic skills and techniques of the related topic. These skills and techniques are important when evaluating the feasibility of the research plan. Then it is important to make a good research plan. Decisions such as what to implement and how to evaluate the research should be made. The trade off between feasibility and creativity should also be done in the research plan. All the decisions and trade off in the plan should be clear, so I will not be lost in the sea of academic papers and user manuals. Otherwise, I can think reversely during self-reflection of my research. If I am lost in the academic papers or user manuals, the reason might be that the research plan is not clear. Unclear research plan may come from lack of knowledge of the related skills and techniques.

6.3 Future work

The experiment in Chapter 3 has proven that it is possible to establish a high-bandwidth network interface between the servers and the optical network. However, the FPGA and the optical transceiver are on separate boards in the current set-up. At the next step, efforts can be placed toward integrating them into a single board.

In Chapters 4 and 5, the FPGA devices are treated as independent processing units in the cloud computing. Therefore, methods that try to manage FPGA resources on their own have been suggested. However, FPGA devices also often run as accelerators in addition to general purpose processors. Existing hardware and software designs for interfaces between FPGAs and processors are based on the system-on-chip architecture where processors and FPGA modules are integrated on the same chip. However, having dedicated servers for a specific type of resource is the key idea of resource disaggregation and helps increase resource efficiency. Similar to the benefits of the disaggregation of memory resources, locating FPGA resources on independent servers can also increase the efficiency of FPGA resources. Therefore in the next step, efforts can be put into developing a communication structure that interfaces FPGA resources with traditional CPU and memory resources through a data centre network in a cloud environment.

BIBLIOGRAPHY

[1] R. Birke, L. Y. Chen, and E. Smirni, “Data centers in the wild: A large performance study.”

[2] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, “Heterogeneity and dynamicity of clouds at scale: Google trace analysis,” in *Proceedings of the Third ACM Symposium on Cloud Computing*, ser. SoCC ’12. New York, NY, USA: ACM, 2012, pp. 7:1–7:13. [Online]. Available: http://doi.acm.org/10.1145/2391229.2391236

[3] Chinese government calls for green data center catch-up. [Online]. Available: https://direct.datacenterdynamics.com/news/chinese-government-calls-for-green-data-center-catch-up/

[4] Guideline for pilot projects of green data centers. [Online]. Available: http://www.miit.gov.cn/n1146295/n1652858/n1653018/c3780626/content.html

[5] A. Shehabi, S. J. Smith, D. A. Sartor, R. E. Brown, M. Herrlin, J. G. Koomey, E. R. Masanet, N. Horner, I. L. Azevedo, and W. Lintner, “United states data center energy usage report,” Tech. Rep., 06/2016 2016.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing.”

[7] R. P. Goldberg, “Survey of virtual machine research,” *Computer*, vol. 7, no. 6, pp. 34–45, June 1974.

[8] R. P. Parmelee, T. I. Peterson, C. C. Tillman, and D. J. Hatfield, “Virtual storage and virtual machine concepts,” *IBM Systems Journal*, vol. 11, no. 2, pp. 99–130, 1972.

[9] L. A. Belady, “A study of replacement algorithms for a virtual-storage computer,” vol. 5, no. 2, pp. 78–101.

[10] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, “Disaggregated memory for expansion and sharing in blade servers,” *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 267–278, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1555815.1555789

[11] W. C. Athas and C. L. Seitz, “Multicomputers: message-passing concurrent computers,” *Computer*, vol. 21, no. 8, pp. 9–24, Aug 1988.

[12] L. Iftode, K. Li, and K. Petersen, “Memory servers for multicomputers,” in *Digest of Papers. Compcon Spring*, Feb 1993, pp. 538–547.

[13] J.-J. Jheng, F.-H. Tseng, H.-C. Chao, and L.-D. Chou, “A novel vm workload prediction using grey forecasting model in cloud data center,” in *The International Conference on Information Networking 2014 (ICOIN2014)*, Feb 2014, pp. 40–45.

[14] A. Khan, X. Yan, S. Tao, and N. Anerousis, “Workload characterization and prediction in the cloud: A multiple time series approach,” in *2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 1287–1294.

[15] L. Chen and H. Shen, “Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters,” in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 1033–1041.

[16] N. Bobroff, A. Kochut, and K. Beaty, “Dynamic placement of virtual machines for managing sla violations,” in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, May 2007, pp. 119–128.

[17] K. Qazi, Y. Li, and A. Sohn, “Workload prediction of virtual machines for harnessing data center resources,” in *2014 IEEE 7th International Conference on Cloud Computing*, June 2014, pp. 522–529.

[18] Z. Xiao, W. Song, and Q. Chen, “Dynamic resource allocation using virtual machines for cloud computing environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, June 2013.

[19] T. Vercauteren, P. Aggarwal, X. Wang, and T. H. Li, “Hierarchical forecasting of web server workload using sequential monte carlo training,” *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1286–1297, April 2007.

[20] W. Fang, Z. Lu, J. Wu, and Z. Cao, “Rpps: A novel resource prediction and provisioning scheme in cloud data center,” in *2012 IEEE Ninth International Conference on Services Computing*, June 2012, pp. 609–616.

[21] K. Mills, J. Filliben, and C. Dabrowski, “Comparing vm-placement algorithms for on-demand clouds,” in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, Nov 2011, pp. 91–98.

[22] Y. Zhang and N. Ansari, “Heterogeneity aware dominant resource assistant heuristics for virtual machine consolidation,” in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 1297–1302.

[23] H. Jin, D. Pan, J. Xu, and N. Pissinou, “Efficient vm placement with multiple deterministic and stochastic resources in data centers,” in *2012 IEEE Global Communications Conference (GLOBECOM)*, Dec 2012, pp. 2505–2510.

[24] J. Xu and J. A. B. Fortes, “Multi-objective virtual machine placement in virtualized data center environments,” in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int’l Conference on Int’l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec 2010, pp. 179–188.

[25] M. Ekman and P. Stenstrom, “A robust main-memory compression scheme,” in *32nd International Symposium on Computer Architecture (ISCA’05)*, June 2005, pp. 74–85.

[26] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro, “Farm: Fast remote memory,” in *Proceedings of the 11th USENIX Conference on Networked Systems Design and*

Implementation, ser. NSDI'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 401–414. [Online]. Available: http://dl.acm.org/citation.cfm?id=2616448.2616486

[27] H. Subramoni, P. Lai, M. Luo, and D. K. Panda, "Rdma over ethernet: A preliminary study," in *2009 IEEE International Conference on Cluster Computing and Workshops*, Aug 2009, pp. 1–9.

[28] P. Faraboschi, K. Keeton, T. Marsland, and D. Milojicic, "Beyond processor-centric operating systems," in *Proceedings of the 15th USENIX Conference on Hot Topics in Operating Systems*, ser. HOTOS'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=2831090.2831107

[29] M. Bielski, C. Pinto, D. Raho, and R. Pacalet, "Survey on memory and devices disaggregation solutions for hpc systems," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, Aug 2016, pp. 197–204.

[30] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker, "Network requirements for resource disaggregation," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 249–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=3026877.3026897

[31] C. Kachris, K. Kanonakis, and I. Tomkos, "Optical interconnection networks in data centers: recent trends and future challenges," *IEEE Communications Magazine*, vol. 51, no. 9, pp. 39–45, September 2013.

[32] G. M. Saridis, S. Peng, Y. Yan, A. Aguado, B. Guo, M. Arslan, C. Jackson, W. Miao, N. Calabretta, F. Agraz, S. Spadaro, G. Bernini, N. Ciulli, G. Zervas, R. Nejabati, and D. Simeonidou, "Lightness: A function-virtualizable software defined data center network with all-optical circuit/packet switching," *Journal of Lightwave Technology*, vol. 34, no. 7, pp. 1618–1627, April 2016.

[33] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[34] K. Katrinis, D. Syrivelis, D. Pnevmatikatos, G. Zervas, D. Theodoropoulos, I. Koutsopoulos, K. Hasharoni, D. Raho, C. Pinto, F. Espina, S. Lopez-Buedo, Q. Chen, M. Nemirovsky, D. Roca, H. Klos, and T. Berends, “Rack-scale disaggregated cloud data centers: The dredbox project vision,” in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 690–695.

[35] G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, “Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation [invited],” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. A270–A285, Feb 2018.

[36] Xilinx, “Vivado design suite user guide UG909 partial reconfiguration,” (Version 2017.1).

[37] J. D. Hadley and B. L. Hutchings, “Design methodologies for partially reconfigured systems,” in *Proceedings IEEE Symposium on FPGAs for Custom Computing Machines*, April 1995, pp. 78–84.

[38] W. Luk, N. Shirazi, and P. Y. K. Cheung, “Compilation tools for run-time reconfigurable designs,” in *Proceedings. The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines Cat. No.97TB100186)*, April 1997, pp. 56–65.

[39] J. Burns, A. Donlin, J. Hogg, S. Singh, and M. D. Wit, “A dynamic reconfiguration run-time system,” in *Proceedings. The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines Cat. No.97TB100186)*, April 1997, pp. 66–75.

[40] W. Fu and K. Compton, “An execution environment for reconfigurable computing,” in *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’05)*, April 2005, pp. 149–158.

[41] A. Ismail and L. Shannon, “Fuse: Front-end user framework for o/s abstraction of hardware accelerators,” in *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2011, pp. 170–177.

[42] D. Koch and J. Torresen, “Fpgasort: A high performance sorting architecture exploiting run-time reconfiguration on fpgas for large problem sorting,” in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA ’11. New York, NY, USA: ACM, 2011, pp. 45–54. [Online]. Available: http://doi.acm.org/10.1145/1950413.1950427

[43] D. Unnikrishnan, R. Vadlamani, Y. Liao, J. Crenne, L. Gao, and R. Tessier, “Reconfigurable data planes for scalable network virtualization,” *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2476–2488, Dec 2013.

[44] C. Dennl, D. Ziener, and J. Teich, “On-the-fly composition of fpga-based sql query accelerators using a partially reconfigurable module library,” in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, April 2012, pp. 45–52.

[45] J. Becker, M. Hubner, G. Hettich, R. Constapel, J. Eisenmann, and J. Luka, “Dynamic and partial fpga exploitation,” *Proceedings of the IEEE*, vol. 95, no. 2, pp. 438–452, Feb 2007.

[46] C. Claus, J. Zeppenfeld, F. Muller, and W. Stechele, “Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system,” in *2007 Design, Automation Test in Europe Conference Exhibition*, April 2007, pp. 1–6.

[47] N. Zilberman, P. M. Watts, C. Rotsos, and A. W. Moore, “Reconfigurable network systems and software-defined networking,” *Proceedings of the IEEE*, vol. 103, no. 7, pp. 1102–1124, July 2015.

[48] L. Devaux, D. Chillet, S. Pillement, and D. Demigny, “Flexible communication support for dynamically reconfigurable fpgas,” in *2009 5th Southern Conference on Programmable Logic (SPL)*, April 2009, pp. 65–70.

[49] T. Becker, M. Koester, and W. Luk, “Automated placement of reconfigurable regions for relocatable modules,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 3341–3344.

[50] R. Pellizzoni and M. Caccamo, “Real-time management of hardware and software tasks for fpga-based embedded systems,” *IEEE Transactions on Computers*, vol. 56, no. 12, pp. 1666–1680, Dec 2007.

[51] A. Rodriguez, J. Valverde, E. de la Torre, and T. Riesgo, “Dynamic management of multi-kernel multithread accelerators using dynamic partial reconfiguration,” in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, May 2014, pp. 1–7.

[52] J. C. Lyke, C. G. Christodoulou, G. A. Vera, and A. H. Edwards, “An introduction to reconfigurable systems,” *Proceedings of the IEEE*, vol. 103, no. 3, pp. 291–317, March 2015.

[53] J. Yang, L. Yan, L. Ju, Y. Wen, S. Zhang, and T. Chen, “Homogeneous noc-based fpga: The foundation for virtual fpga,” in *2010 10th IEEE International Conference on Computer and Information Technology*, June 2010, pp. 62–67.

[54] 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 *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, June 2014, pp. 13–24.

[55] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, D. Firestone, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Y. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, “Configurable clouds,” *IEEE Micro*, vol. 37, no. 3, pp. 52–61, 2017.

[56] 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.

[57] 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

[58] 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.

[59] O. Knodel and R. G. Spallek, “Computing framework for dynamic integration of reconfigurable resources in a cloud,” in *2015 Euromicro Conference on Digital System Design*, Aug 2015, pp. 337–344.

[60] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Y. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, “A cloud-scale acceleration architecture,” in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–13.

[61] 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.

[62] J. Weerasinghe, F. Abel, A. Herkersdorf, and C. Hagleitner, “Disaggregated fpgas: Network performance comparison against bare-metal servers, virtual machines and linux con-

tainers," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2016, pp. 9–17.

[63] N. Tarafdar, N. Eskandari, T. Lin, and P. Chow, "Designing for fpgas in the cloud," *IEEE Design Test*, vol. 35, no. 1, pp. 23–29, Feb 2018.

[64] R. Brodersen, A. Tkachenko, and H. K. H. So, "A unified hardware/software runtime environment for fpga-based reconfigurable computers using borph," in *Proceedings of the 4th International Conference on Hardware / Software Codesign and System Synthesis (CODES+ISSS '06)*, Oct 2006, pp. 259–264.

[65] C. Chang, J. Wawrzynek, and R. W. Brodersen, "Bee2: a high-end reconfigurable computing system," *IEEE Design Test of Computers*, vol. 22, no. 2, pp. 114–125, March 2005.

[66] 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.

[67] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402967

[68] "Cisco global cloud index: Forecast and methodology, 2014 to 2019 white paper." [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html

[69] R. Courtland, "Can hpe's "the machine" deliver?" *IEEE Spectrum*, vol. 53, no. 1, pp. 34–35, January 2016.

[70] S. Han, N. Egi, A. Panda, S. Ratnasamy, G. Shi, and S. Shenker, "Network support for resource disaggregation in next-generation datacenters," in *Proceedings of the Twelfth*

ACM Workshop on Hot Topics in Networks, ser. HotNets-XII. New York, NY, USA: ACM, 2013, pp. 10:1–10:7. [Online]. Available: http://doi.acm.org/10.1145/2535771.2535778

[71] J. Glanz, "Google details electricity usage of its data centers." [Online]. Available: https://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html

[72] "Software-defined networking (SDN) definition." [Online]. Available: https://www.opennetworking.org/sdn-definition/

[73] "Network function virtualization." [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[74] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[75] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. Berkeley, CA, USA: USENIX Association, 2015, pp. 117–130. [Online]. Available: http://dl.acm.org/citation.cfm?id=2789770.2789779

[76] "Network functions virtualisation white paper 1." [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[77] H. Koumaras, C. Sakkas, M. A. Kourtis, C. Xilouris, V. Koumaras, and G. Gardikis, "Enabling agile video transcoding over sdn/nfv-enabled networks," in *2016 International Conference on Telecommunications and Multimedia (TEMU)*, July 2016, pp. 1–5.

[78] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[79] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[80] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 655–685, Firstquarter 2016.

[81] "Network functions virtualisation white paper 3." [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf

[82] C. H. Huang and P. A. Hsiung, "Hardware resource virtualization for dynamically partially reconfigurable systems," *IEEE Embedded Systems Letters*, vol. 1, no. 1, pp. 19–23, May 2009.

[83] 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.

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

[85] 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.

[86] M. K. Papamichael and J. C. Hoe, "Connect: Re-examining conventional wisdom for designing nocs in the context of fpgas," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12. New York, NY, USA: ACM, 2012, pp. 37–46. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145703

[87] M. Papamichael, "CONNECT: CONfigurable NEtwork creation tool." [Online]. Available: http://users.ece.cmu.edu/~mpapamic/connect/

[88] S. C. V. A. Group, "Enabling technology for on-chip networks." [Online]. Available: http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/WikiStart

[89] A. Monemi, J. W. Tang, M. Palesi, and M. N. Marsono, "Pronoc: A low latency network-on-chip based many-core system-on-chip prototyping platform," *Microprocessors and Microsystems*, vol. 54, pp. 60 – 74, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933117302417

[90] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "Netfpga sume: Toward 100 gbps as research commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, Sept 2014.

[91] 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.

[92] K. Lim, Y. Turner, J. R. Santos, A. AuYoung, J. Chang, P. Ranganathan, and T. F. Wenisch, "System-level implications of disaggregated memory," in *IEEE International Symposium on High-Performance Comp Architecture*, Feb 2012, pp. 1–12.

[93] A. D. Papaioannou, R. Nejabati, and D. Simeonidou, "The benefits of a disaggregated data centre: A resource allocation approach," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–7.

[94] R. Hou, T. Jiang, L. Zhang, P. Qi, J. Dong, H. Wang, X. Gu, and S. Zhang, "Cost effective data center servers," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 179–187.

[95] "PEX 8796 product overview (563 KB)." [Online]. Available: https://docs.broadcom.com/docs/12351860

[96] "polatis - optical switch modules from 4x4 up to 192x192 ports." [Online]. Available: http://www.polatis.com/

switch-modules-for-oem-all-optical-switch-module-solutions-original-equipment-manufactures.asp

[97] M. Vestias and H. Neto, “Trends of cpu, gpu and fpga for high-performance computing,” in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Sept 2014, pp. 1–6.

[98] Y. Wang, J. Yan, X. Zhou, L. Wang, W. Luk, C. Peng, and J. Tong, “A partially reconfigurable architecture supporting hardware threads,” in *2012 International Conference on Field-Programmable Technology*, Dec 2012, pp. 269–276.

[99] G. Zervas, F. Jiang, Q. Chen, V. Mishra, H. Yuan, K. Katrinis, D. Syrivelis, A. Reale, D. Pnevmatikatos, M. Enrico, and N. Parsons, “Disaggregated compute, memory and network systems: A new era for optical data centre architectures,” in *2017 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2017, pp. 1–3.

[100] “Xilinx zynq UltraScale+ MPSoC ZCU102 evaluation kit.” [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html