Front. Comput. Sci., 2023, 17(5): 175106
https://doi.org/10.1007/s11704-022-2127-0

REVIEW ARTICLE

FPGA sharing in the cloud: a comprehensive analysis

Jinyang GUO[1], Lu ZHANG[1], José ROMERO HUNG[2], Chao LI (✉)[1], Jieru ZHAO[1], Minyi GUO[1]

1 Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
2 Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai 200240, China

Abstract Cloud vendors are actively adopting FPGAs into their infrastructures for enhancing performance and efficiency. As cloud services continue to evolve, FPGA (field programmable gate array) systems would play an even important role in the future. In this context, FPGA sharing in multi-tenancy scenarios is crucial for the wide adoption of FPGA in the cloud. Recently, many works have been done towards effective FPGA sharing at different layers of the cloud computing stack.

In this work, we provide a comprehensive survey of recent works on FPGA sharing. We examine prior art from different aspects and encapsulate relevant proposals on a few key topics. On the one hand, we discuss representative papers on FPGA resource sharing schemes; on the other hand, we also summarize important SW/HW techniques that support effective sharing. Importantly, we further analyze the system design cost behind FPGA sharing. Finally, based on our survey, we identify key opportunities and challenges of FPGA sharing in future cloud scenarios.

Keywords cloud FPGA, FPGA sharing, efficiency, design cost

1 Introduction

We are now entering a booming era of cloud computing. There is a growing enthusiasm for pursuing sustainability in cloud scenarios. Cloud service providers (CSPs) [1−5] integrate FPGAs into data centers for improving performance and efficiency. Under continuous evolution of cloud service, FPGA-based computing would play an even more important role in the future.

Researchers and developers are actively dashing into the region for enhancing the efficiency of cloud FPGA. Some surveys summarize related art with specific perspectives. As shown in Table 1, these reviewing works pay attention to various aspects of cloud FPGA. Some of them [6,7] concern about the applicability, and the ideal combination between the cloud FPGAs and typical tasks. Other like [8] discusses the trend of the architecture of future cloud FPGA. Quraish et al. [9] review the techniques of FPGA virtualization. A few works [10−13] focus on the security issues of cloud FPGA.

Pre-existing surveys in literature lack adequate discussion over the important topic of FPGA sharing. FPGA sharing in multi-tenancy scenarios is crucial for the wide adoption of FPGA in the cloud. These works mainly focus on a specific optimization perspective of cloud FPGA. In recent years, a considerable amount of works have been done towards effective FPGA sharing at different layers of the cloud computing stack. Comprehensive exploration of prior works in the topic would allow a better cost-effect analysis over cloud FPGA multi-shared implementation.

In this work, we provide a comprehensive survey on FPGA sharing in this work. We examine prior art from different aspects and encapsulate relevant proposals on a few key topics. Firstly, we discuss representative papers on FPGA resource sharing. We categorize related works based on the organization of FPGA resources. Then, we summarize important software and hardware techniques that support FPGA sharing from prior works. Importantly, we take a further step to analyze the system design cost of FPGA sharing. We summarize the related design costs and categorize them according to sources and processing stages.

The intended audiences of this work are: 1) The CSPs who want to introduce new FPGA-instances depending on sharing schemes. 2) Developers and researchers who are interested in cloud-based FPGA systems in multi-tenancy scenarios.

We make several key contributions as follows:

- We examine the related works of resource-oriented FPGA sharing. We discuss corresponding concepts of shared resources and provide a classification based on the hierarchical organization.
- We classify research works according to FPGA sharing supports including hardware support, software support, and security guarantee. We describe major system optimization solutions.
- We present a preliminary cost study of sharing-based FPGA systems. Focusing on the attribution and processing stages, we classify these connected expenses.

The organization of this work is shown in Fig. 1. Section 2 provides an overview of FPGA sharing. Section 3 discusses

Received March 7, 2022; accepted July 25, 2022

E-mail: lichao@cs.sjtu.edu.cn

Table 1 Other surveys of the cloud FPGA in multi-tenancy scenarios

Related survey works	Focus	Year
Survey of deep learning neural networks implementation on FPGAs [6]	Application	2020
Accelerating DNNs from local to virtualized FPGA in the Cloud: a survey of trends [7]	Application	2021
Survey and future trends for FPGA cloud architectures [8]	Architecture	2021
A survey of system architectures and techniques for FPGA virtualization [9]	Virtualization	2021
Security of FPGAs in data centers [10]	Security	2017
Trust in FPGA-accelerated cloud computing [11]	Security	2021
Security of cloud FPGAs: a survey [12]	Security	2021
SoK: secure FPGA multi-tenancy in the cloud: challenges and opportunities [13]	Security	2021

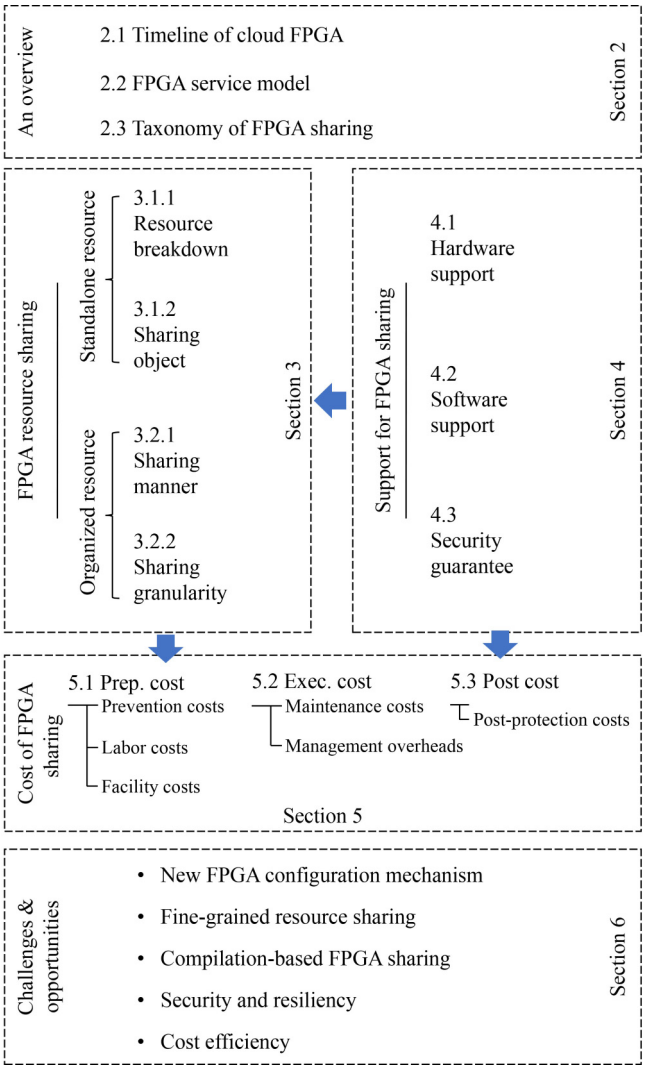


Fig. 1 The organization of this work

resource-sharing schemes. Section 4 summarizes the existing supports for FPGA sharing. Section 5 presents the cost analysis as a supplement. Section 6 gives a discussion of the challenges and opportunities. Finally, we make the conclusion in Section 7.

2 An overview: FPGA sharing

The call for flexibility and sustainability makes FPGA stand out among optional processing technologies. There is a mega-trend for cloud vendors integrating FPGA into their infrastructures. Mainstream cloud service providers (CSPs) [1−5] adopted this programm-able device for improving the efficiency of cloud computing in the last decade. We provide a brief overview of cloud FPGA in this section.

2.1 Timeline of cloud FPGA

The history of cloud FPGA may be traced back around a decade. Modern FPGAs gain improvement in heterogeneity and specific acceleration due to the development of techniques. Researchers have begun to pay attention to the FPGA in the cloud [14]. Microsoft has announced their public FPGA project Catapult [15], and presented a CPU-FPGA collaborative design as its acceleration core unit.

Following this, Intel has started a FPGA project called hardware acceleration research program (HARP) [16]. They put forward an emerging technology called partially reconfiguration by dividing the hardware design into two different bitstreams, the static part, and the dynamic part. The partial reconfiguration allows the configuration of FPGA to be more efficient in the cloud and lays a foundation for the fine-grained sharing of FPGA.

Shortly afterward, the public cloud platforms complied with the trend of FPGA cloudification and released their FPGA-instances [1−5]. These CSPs sell the FPGA-equipped instances in the granularity of the board. The sharing of FPGA mainly stays in the status of coarse-grained. This monotonous formulation diminishes the efficiency of both economy and programming flexibility.

Recently, researchers explore on finer-grained sharing of FPGA. Some works introduce new abstractions of the hardware resources of FPGAs [17−19]. There also several works [20−22] providing detailed analyses over FPGA hardware hierarchy for enhancing effective FPGA sharing.

2.2 FPGA service model

There are a tremendous amount of works [23−26] proving the strengths of FPGA in performance and efficiency. CSPs can utilize FPGAs for improving the flexibility and sustainability of their cloud services. In different cloud service models, CSPs can provide various FPGA-oriented computing service model for the requirements of users. Specifically, the FPGA-based computing service model can be divided into: 1) FPGA in IaaS [1−3,19]. CSPs provide the computing service with whole FPGA boards. The devices react as unconfigured computing resources for customization. 2) FPGA in PaaS [4,5]. CSPs provide the FPGA service by configured development platforms. 3) FPGA in SaaS [15]. CSPs provide the FPGA as a defined accelerator. Users can utilize the pre-equipped function with specific APIs.

Additionally, some emerging FPGA-native service models

are also discussed in the literature. FPGA can be regarded as the independent form of the computing service. BlastFunction [27] and Fasten [28] extend serverless computing onto FPGA and find a new model called FPGA-as-a-service (FaaS). Skhiri et al. [29] propose an intermediate expression of the hardware design which defines a new model as the IP-as-a-Service (IPaaS).

2.3 Taxonomy of FPGA sharing

We have witnessed a growing interest in cloud-based FPGA systems. FPGA sharing in multi-tenancy is the key for wide adoption of FPGA in the cloud [18,30,31]. The usage of cloud FPGA tends to be more flexible with corresponding architectures [8].

Mechanism aside, there are still some commonalities of FPGA sharing in different service models. We summarize these similarities from the aspects of resource sharing schemes, sharing supports, and system design costs as shown in Fig. 2. In the later section, we will discuss FPGA sharing in detail based on the taxonomy.

3 FPGA resource sharing

Modern FPGAs are complex devices with a considerable variety of integrated technologies and resources. Many literature works present their concerns over the sharing of FPGA resources. In this section, we discuss different schemes of FPGA sharing according to specific resources.

This section presents two complementary parts based on the resource organization. We will discuss FPGA sharing strategies based on resource organization. Prior art of resource-oriented FPGA sharing is divided into two types, standalone resource sharing and organized resource sharing. Standalone resource sharing and organized resource sharing have distinct issues. Initial FPGA resources without organizations are known as standalone resources. Organized resources are a subset of FPGA resources that are grouped into partitions.

3.1 Standalone resource sharing

The works of FPGA sharing based on the standalone resource concerns of the architecture features of the FPGAs. The hardware resources on FPGA board can be categorized based on the programmability [32]. Some FPGA resources are configurable, and the others are static resources. Programmable resources usually gather together and compose as the dynamic region or the role namely. The counterpart

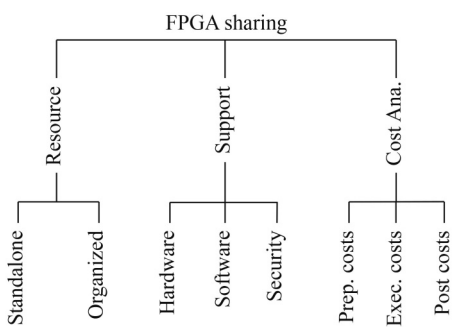


Fig. 2 A taxonomy of FPGA sharing

makes up the static region or the shell. We discuss the resource types and the specific sharing objects to analyze standalone resource sharing. FPGA resources are shared through specific sharing objects.

3.1.1 Resource breakdown

The prior art of FPGA based on the standalone resource pay close attention to the manufacturing [8,20,32−36]. These works enhance programmable configuration of FPGA in hardware features including capacity, connectivity, and storage. We divide the resources of FPGA depending on functions. The resources contain logic resources, connectivity resources, and memory resources.

Logic resource The logic resources of FPGA are used for function customization [20,32,33]. Typical FPGA logic resources contain the functional resources and wiring resources which can be further divided into the pre-defined control logic and programmable units. The pre-defined logic contains function units such as the DSP and clock controller. The programmable units mainly contains the LUT, register, and the configurable I/O. A deeper level of logic sharing involvement have been proposed by advanced LUTs logic such as the bank of centroids represented by Romero Hung et al. [37]. Although not directly tested within a cloud environment, their centroid sharing logic proved effective acceleration features when dynamically shared among independent processing units and while dealing with complex real-world datasets as the intervening in cloud computing.

Connectivity resource FPGA contains several network and connectivity resources [34] for specific purposes. QuickPath Interconnect (QPI) and Coherent Accelerator Processor Interface (CAPI) [8] can be used to guarantee data consistency in the FPGA-CPU shared memory. The sharing of the network resources is mainly based on the bandwidth. For FPGA sharing in multi-tenancy scenarios, some efforts are needed for arranging the requests from the multi-purpose design.

The wiring resources are usually neglected. The optimization of wiring resources is directly related to the use and implementation results. The wiring resources are mainly used to connect all the functional modules. The wiring resources are used for function connection and clock synchronization.

Memory resource The memory resources equipped in FPGAs can be divided into two categories according to the location, on-chip memory and off-chip memory. The on-chip memory is used as temporary storage, and the off-chip is used as data storage. Mostly, on-chip memory is much faster but less available than off-chip memory.

Normal FPGAs contain two types of on-chip memory [35]: Distributed RAM and the Block RAM (BRAM). Xilinx also presents UltraScale serie device equipped with URAM [38]. This memory is a storage with large volume and access efficiency.

The off-chip memory of FPGA is mainly DRAM based storage. Several high-end devices are equipped with emerging storage called high-bandwidth memory (HBM) (e.g., Xilinx Alveo U280 [36] & U50 [32], and Intel Stratix 10 MX [33]).

For FPGA sharing, efficient memory access management is

critical. For memory sharing, Ma et al. [39] use hypervisor-based management. They introduce a functional isolation for the maintenance of accelerators. To achieve memory sharing, Tarafdar et al. [40] create hypervisors in the static section of the FPGA. Another viable approach for sharing on-chip memory exists. AXI-based memory management is presented by Vaishnav et al. [41].

3.1.2 Resource sharing object

Sharing object is virtual resources providing an abstraction for FPGA sharing management. In Sec 3.1.1, we have divided the standalone resources into three categories, logic resources, connectivity resources, and memory resources. As shown in Table 2, the sharing of these resources is complex in categories and configuration. FPGA resources sharing depends on a specific sharing object. We will discuss the FPGA sharing objects with the specific resources as follows.

Configuration sharing Configuration sharing is deployed on the occupation of FPGA hardware. This sharing object is typically oriented to logic resources. Logic resources are shared temporally with configuration updating [42,43]. Works like [52] presents a NoC-based system for run-time reconfiguration. Tenants occupy the control in the rent period with the corresponding configurations.

Bandwidth sharing Bandwidth sharing is based on the co-located traffic control. Sharing degrees should be limited to meet the requirement for efficiency. CSPs need to allocate the co-execution carefully to avoid run out the bandwidth:

1) The migration of computing data will be time-consuming if the network gets congestion.

2) The computing progress will be blocked if the off-chip memory runs out of bandwidth.

Memory and the network resources are shared typically based on the bandwidth. Maany studies pay attention to connections related to the sharing of FPGA. OS4RS [53] presents a hybrid mechanism of communication with the implementation of both hardware and software. VFR [54] treats the FPGA as the network component based on the BSP protocol. Catapult [15] makes a contribution to the host communication and network access. There are also many works concerning the PCIe controller [14,30,44−49] for configuration control.

Capacity sharing Memory resources sharing also depends on the storage capacity. Off-chip memories have sufficient capacity usually. However, the sharing of on-chip memory should be handled carefully due to the limitation of storage capacity [51].

As shown in Fig. 3, each standalone die is connected with memory resources. The cross-die reading will consume the scarce wiring resource for connection. There are some constraints for the die-crossing [36]. We should reduce or even disable the die-crossing design in the sharing FPGA.

3.2 Organized resource sharing

Organized resource sharing is a type of resource sharing that is based on collection of grouped FPGA resources. Die-stacking technique is used in existing FPGAs in the cloud [1−5]. This architecture adds a layer of complexity to the management of FPGA resources. FPGA devices may be thought of as a collection of sub-devices. These sub-devices are made up of separate dies. This organization of FPGAs contains separate logic resources, memory resources, and connectivity for a monolithic die.

FPGA vendors release high-end devices with the trade-off

Table 2 FPGA resources and the sharing object

Resource category	Resource type	Configuration & description	Sharing object
Logic resource [42,43]	Programmable I/O unit	Configurable I/O port	Configuration
	Programmable logic unit	LUT and register	
	Function unit	Pre-defined hardware functions	
	Hard core	Control hard core	
Connectivity [14,30,44–49]	PCIe IP	PCI express control	Bandwith
	NIC	Network control	
	Wiring resource	Configurable connectivity units	
Memory [15,39,50,51]	Distributed RAM	Multi-mode memory	Capacity & bandwith
	DRAM	Off-chip memory storage	
	HBM	High bandwidth storage	
	BRAM	RAM with changeable port	

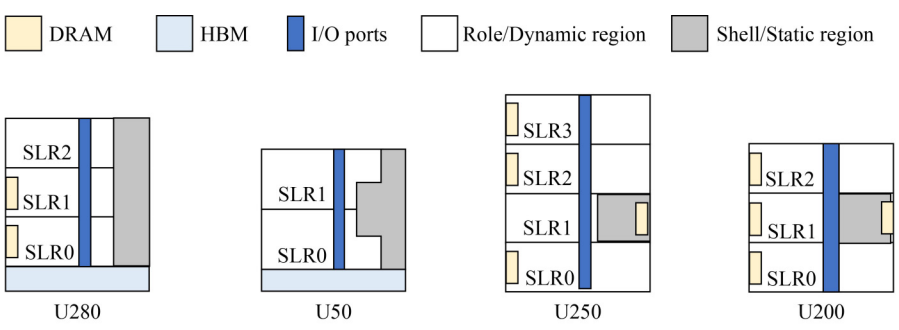


Fig. 3 Floorplans of Xilinx Alveo series boards

between capacity and economy [32]. Figure 3 illustrates the floorplans of Xilinx Alveo series. These devices are multi-die based and each monolithic die is called Super Logic Region (SLR). The stacking of the dies introduces natural hardware boundaries. Each die is an independent unit for resource management [55]. Further, several works [21,45,47,54,56] provide a finer-grained FPGA units. They introduce a new resource collection called partial reconfigurable region (PRR).

We regard FPGA sharing as a unique type of hardware sharing flexible hierarchical organization. In other words, it has multiple granularity (the board, the die, and the PRR). Firstly, we discuss FPGA sharing with temporal hardware multiplexing. Secondly, we examine prior art on spatial sharing of FPGA hardware resources.

3.2.1 FPGA sharing granularity

FPGA sharing has several levels of granularity depending on the physical organization structure. The granularity of FPGA sharing is summarized in Fig. 4 as board-level sharing, die-level sharing, and PRR-level sharing.

Board-level sharing The sharing of the board-level includes two forms, multi-board sharing and single-board sharing [1−3,5,15]. The implementation of board-level sharing presents as the exclusive occupation of the control during use.

Die-level sharing The sharing of the die-level is a kind of fine-granularity spatial sharing. The device is divided into the sub-devices of chips [20,22]. Users deploy their designs on different dies by co-locating their functions on the same board and accomplish their functions independently.

PRR-level sharing The sharing of PRR-level is finer-grained than the die-level. Users co-locate their hardware design inner a die area [21]. This sharing level [18,31,46] requires users to have a preliminary portrayal for resource utilization. There are two types of PRR-level sharing, depending on the dynamic properties. Specifically, PRR-sharing can be divided into static PRR-sharing and dynamic PRR sharing. We mark the former type as the SPRR and the latter as the DPRR.

3.2.2 FPGA sharing manner

The technique for hardware multiplexing is defined as the sharing manner. In multi-tenancy contexts, FPGA sharing primarily uses two types of multiplexing: time-division multiplexing (TDM) and space division multiplexing (SDM). Temporal sharing is possible with the TDM design, while spatial sharing is possible with the SDM design.

The flexibility of sharing implementation is ensured through manners. These FPGA-equipped instances can be used effectively by tenants. As a result of these sharing mechanisms, tenants will prefer cost-effective FPGA computing.

TDM The mechanism of TDM is used to share FPGAs temporally. By lowering the TCO of FPGAs, FPGA temporal sharing enables competitive cloud computing [14,56−58]. The specific implementations are mainly based on configuration rotation. The CSPs rely on the exchange of FPGA settings to implement new functionality. These TDM designs concern the scheduling of the functions due to the reconfiguration.

SDM The mechanism of SDM enables FPGAs can be

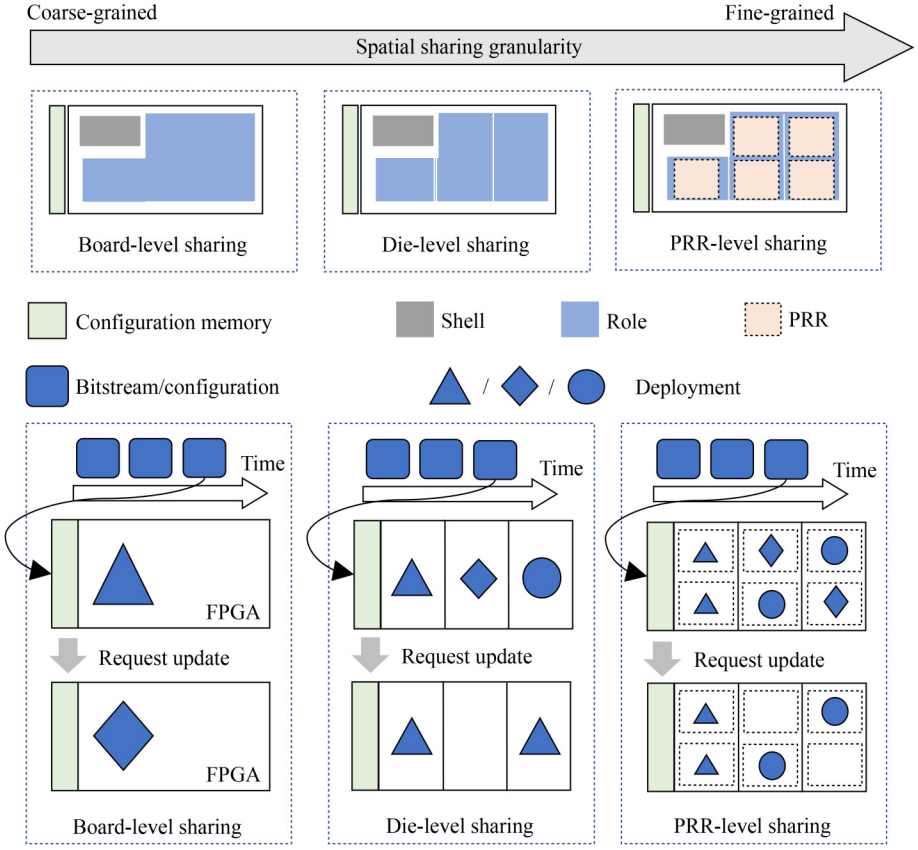


Fig. 4 FPGA sharing in multi-granularity

shared spatially.FPGA spatial sharing [18,21,31,59−62] offers lower selling costs on a small scale. SDM based on the flexible partitions of FPGA. The spatial sharing allows different functional designs to co-execute on the same FPGA board. Several academic works [18,21,31,59−62] explore this region. They present the mechanism of FPGA virtualization [18,31] and the pooling technique [21].

4 Support for FPGA sharing

Flexible FPGA sharing requires specific systems and techniques for support. In this section, we examine the related supporting art and categorize the supporting techniques into three categorises, hardware support, software support, and security guarantee as shown in Table 3.

4.1 Hardware support

FPGA sharing tends to be diverse in manners and granularity. The flexible FPGA sharing is based on the hardware multiplexing. FPGA device multiplexing requires several basic hardware supports: 1) reconfiguration, 2) connectivity, and 3) partition.

Reconfiguration Users can update hardware configurations with their own hardware designs. The devices are programmed with bitstreams loading in the configuration memory. Thus, Reconfiguration is the key hardware support for FPGA sharing [32,33]. FPGA could be shared with different configurations from tenants. The reconfiguration is mainly based on the programmability of FPGA.

Efficient programmability endows FPGA with the customization of functions. Reconfiguration [63,81] is the crucial supporting guarantee for the design updating. The techniques of partial reconfiguration [15,21] supports the FPGA sharing spatially with hardware updating. Especially, the support for partial-reconfiguration is the key to improve the flexibility of FPGA sharing in cloud multi-tenancy scenarios [64,66,67]. Partial reconfiguration enables the fine-granular sharing.

Connectivity FPGA sharing in multi-tenancy manifests as functions co-locating on the boards. Connectivity is important to support the hardware designs for function co-location. FPGA is equipped with abundant resources for connection [30,46] including wiring sources, PCIe and NICs. The convenient connectivity allows to co-locate hardware designs for FPGA spatial sharing.

Partition Effective cloud-based FPGA systems usually target on the specific devices [1,21,55]. These devices support various partition of resources which can organize resources as hierarchical collections. These independent collections improve FPGA sharing flexibility with multi-granularity which we have discussed in section 3.2.1. Board-level sharing, die-level sharing, and PRR-level sharing are all predicated on how FPGA resources are partitioned. For fine grained FPGA sharing, the die-level sharing relies on the partition of the stacked chip, and the PRR-sharing is based on a finer resource organization inner dies.

Table 3 Supports for FPGA sharing

Category	Support	Related work
Hardware	Reconfiguration	[15,21,32,33,63]
	Connectivity	[30,46]
	Partition	[1,21,55]
Software	Isolation	[64]
	Migration	[65,66]
	Abstraction	[64,67]
	Virtualization	[18,68,69]
	Management	[15,46,60]
	Scheduling	[21,44,47,56,69]
	Development	[18,31,47]
Security	Data Confidentiality	[70–72]
	Bitstream Protection	[60,73–75]
	Power Control	[76–78]
	Workload Isolation	[3,30,45,79]
	Configuration Refresh	[63,80]

4.2 Software support

Plenty of prior papers study the software support for FPGA sharing in multi-tenancy scenarios which are from both academia [14,18,30,31] and industry [15,82]. To support FPGA sharing, researchers design cloud-based FPGA systems for efficient resource management and hardware abstraction. As shown in Table 4, prior art contributes wisdom of software supports over the aspects of performance isolation, task migration, software abstraction, resource virtualization, resource management, tasks scheduling and development tools.

Performance isolation Efficient schemes of isolation are necessary for concurrent execution of tasks. Casual designs will make the FPGA clusters suffer from the performance degradation of the interference. Feniks system [64] for the cloud FPGA ensure the performance with the isolation of application region and OS region.

Task migration The live-migration of FPGA tasks helps improving resources utilization in the cluster wide. Knodel et al. [65] import a migration scheme for long-run task using virtualization. Vaishnav et al. [66] introduce a technique for lively task migration and can help the FPGA cluster benefiting from the overlapping the computation and data movement.

Software abstraction The abstraction of software improves the efficiency of FPGA sharing development. The work of Coyote system [67] imports OS abstractions to FPGA and supports secure spatial and temporal multiplexing. Fenicks system [64] provides abstracted I/O interfaces for convenient development to accelerators in PRRs.

Resource virtualization Several works provide flexible virtualization. These works can be divided into three-level. The first level only provides limited virtualization for I/O reuse [82,84] but omitted the virtualization of logic resources. The second level introduces the abstraction of both the I/O and logic resources [18,21,68]. There are the other special types [60,69] for presenting application-oriented virtualization. They mainly provide an API for calling the pre-defined accelerators which can be seen as the extension of SaaS Model.

Resource management There are a few works on the management of FPGA resources. In these works, the management of the FPGA is various due to the controlling and the sharing objects. But these work similarly provide the abstraction for the hardware resource management. According

Table 4 Software support of FPGA sharing, categorized by target partition of device

Work	SW support	Organization	Contorl unit	Sharing manner	Year	Key technique
RC3E [56]	RV, RM, TS	PRR	I/O	S&T	2016	Hypervisor
VFR [54]	RV, RM, TS	PRR	I/O	S&T	2016	PRR manager
RRaaS [45]	RV, RM, TS, DT	PRR	API	S&T	2016	NoC, hypervisor
hCODE [47]	RV, RM, TS, DT	PRR	I/O & logic	S&T	2018	PRR manager
FPGApooling [21]	RV, RM, TS	PRR	I/O& logic	S&T	2021	Pooling scheduler
Feniks [64]	RV,RM,SA,PI	DPRR	I/O & logic	S&T	2017	Region isolation, I/O abstraction
RACOS [46]	RV, RM	DPRR	Accelertor API	S&T	2017	PCIe controller
LiveMig [66]	RV,TM,TS	DPRR	I/O& logic	S&T	2018	Live task migration
AmorphOS [30]	RV, RM	DPRR	I/O	S&T	2018	PCIe controller, Zone manager
ViTAL [18]	RV, RM, DT	DPRR	I/O & logic	S&T	2020	Compiler, Hypervisor
Coyote [67]	RV, RM, SA, TS	DPRR	I/O & logic	S&T	2020	OS abstraction
Hetero-ViTAL [31]	RV, RM, TS, DT	DPRR	I/O & logic	S&T	2021	Hypervisor, LL Block
pvFPGA [44]	RM, TS	Board	Accelertor API	T	2013	Xen VMM
Catapult [15]	RM, TS	Board	I/O & Network	T	2014	Resource controller
VFACC [60]	RM, TS	Board	Accelertor API	T	2015	Hypervisor, PCIe controller
Blaze [57]	RM, TS, DT	Board	I/O	T	2016	Run-time manager
QMC [83]	RM, TS	Board	I/O & Network	T	2017	Avalone connector
Migration [65]	RM,TN,TS.	Board	API & Network	T	2017	vFPGA-based Migration
RTDNN [69]	RM, TS	Board	Accelertor API	T	2018	Hypervisor
FPGAvirt [49]	RM, TS	Board	Network	T	2018	NoC, Sevice manager

* PI means performance isolation, TM means task migration, SA means software abstraction, RV means resource virtualization.
RM means resource management, TS means task scheduling, DT means development tools.

to the categories of the resources, these works can be divided into three types. Specifically, these techonologies contains logic resource management [47,54], connectivity resource management [15,46,49,60,83], and memory resource management [21,39].

Task scheduling The scheduling is an important issue for the requirement in multi-tenancy scenarios. Some works [44,56,69] extend the mechanism of VM, and provide their scheduling strategies. There are also several emerging scheduling-based techniques. Zhao et al. presented an accelerator oriented controller and scheduler based on the hCODE system [47]. Zhu et al. [21] provided a scheduler for FPGA resource pooling.

Development tools Some works extend the existing software stack of the cloud computing [44,45,47]. Other works present their unique designs. ViTal [18] and the hetero-Vital [31] provide emerging compiling tools with the block-based abstraction of hardware and software. Blaze [57] presents a run-time manager by implementing the controller on the static region of the FPGA.

Furthermore, prior art based on FPGA sharing may be separated into two categories: works for general purpose and works for domain-specific applications. Many FPGA sharing-related works are carried out for a specific purpose. There are some works designed for the general purpose. AmorphOS [30] shows the interests in the wide field, including CNN and Bitcoin. FPGAvirt [49] presents an all-purpose NoC. VFR [54] shows the concern for the efficiency of the compiler. There are a few works focusing on the application for specific domains. A series of works [18,31] present their wisdom in the region of machine learning. Some works also take part in the popular realm of graph processing [15,68].

4.3 Security guarantee

Security is the concerning problem with priority for FPGA sharing in multi-tenancy scenarios. CSPs and users would suffer losses due to potential threats. FPGA sharing not only involves traditional issues of hardware and software, but the security guarantee is also important. Threats from open programming are severe. Researchers and developers extensively pay attention to the security problem of FPGA sharing [10,13,85]. The common threats include network strikes, hardware attacks, software attacks, and assets leaks.

As shown in Table 3, the corresponding methods handling these threats include but are not limited to the workload isolation [3,30,45,79], bitstream protection [60,73−75], power control [76,86], and configuration refresh [76,86]. Additionally, security issues vary from sharing granularity. The board-level sharing mainly concerns configuration protection. FPGA boards are shared with the rotation of the hardware occupation. The configuration memory of FPGA should be emptied to avoid the ghost circuit [73]. As for finer-grained sharing, the management of FPGA needs to be handled carefully. There are severe threats to the co-execution of tasks. As countermeasures, effective power control [63,80] and function isolation [10,74,87] are required.

Data confidentiality Data confidentiality is crucial in the virtualized FPGA sharing virtualized. Yazdanshenas et al. [70,71] focus on this issue and import a data protection by encrypting/decrypting users' application. Zhao et al. [72] also provide similar security measures for protecting the FPGA computing in the cloud.

Bitstream protection The protection of FPGA bitstreams is a preventive solution for defending the potentially destructive. State-of-the-art FPGA sharing systems provide bitstream protection with data verification [73,74]. Bitstreams validation can guarantee the designs are harmless and complete. Specific implementations for bitstream validation contain the methods of DRCs and virus scanners. However, these methods require access to the client IP designs, which would introduce IP conflicts [13].

There are also other effective methods for FPGA bitstreams protection. As an intermediate of the bitstream, IPs are also important. To protect IPs, SeRFI [75] introduces an obfuscate mechanism with a finite state machine to protect IP. There are a number of encryption-based security options. VFACC [60] uses encryption to validate memory accesses and only permits legitimate memory accesses.

Power control Power control acts as countermeasures for power-related threats. The malicious tenant can leverage the power leakage to construct a covert channel between malicious co-tenants [55]. FPGA sharing in multi-tenancy scenarios suffers for the power leakage attacks. Prior art provide solutions including voltage-based active fences [76], the run-time frequency tuning [77], and network of RO-based sensors for attack detection [78].

Workload isolation Workload isolation is used to defense the software attacks in execution. There are potential malicious co-tenants stealing or even damaging the data [10]. Existing cloud-based FPGA systems [3,30,45,79] often implement mechanisms to isolate applications and to randomise this isolation. The isolation enhances the independence of data and hardware design. CSPs can then provide clients with sufficient guarantees that their assets and data remain contained in secure without manipulation during processing.

Configuration refresh The configuration refresh guarantee the accuracy of the configuration updating [32,33]. Prior art adopts this scheme to avoid the ghost circuit and design leaks. FPGA is configured with the bitstream. The function of the FPGA hardware updates through bitstream downloading into the configuration memory. Configuration refresh can avoid ghost circuit without the false implementation [63,80]. Importantly, emptying the FPGA configuration can protect tenants from assets leaks in the release stage.

5 Cost of FPGA sharing

We categorize the cost-related works, and summarize the parameters of the cost issues. As shown in Table 5, we divided the processing into stages and summarize related costs. There are three processing stages including preparation, execution and release. Each stage generates their own costs.

We discuss related works of FPGA sharing technologically in the previous sections. The cost problem is also an open region for research in cloud computing [91]. FPGA sharing impacts the costs from several aforementioned aspects. In this section, we analyze the cost of FPGA sharing.

FPGA sharing costs can be classified based on the processing stage, including preparation costs, execution costs, and post costs (such as costs in releases). The discrepancies in quantification may be seen in these FPGA sharing-related charges. Money costs can be quantified for some costs that are present in the form of expenditure capital. The other manifests as overhead and consumption, and their measurement is linked to the length of time and occupation. These costs includes FPGA idle time, energy consumption, safety maintenance, and performance overhead are all included in these expenses. Despite the fact that these costs are not direct commercial expenses, they may be transformed to a unified form of money cost depending on the duration and consumption.

5.1 Preparation costs

Preparation costs are generated in the preparation. These costs are mainly expressed as the consumption of the money. Typically, these costs include facility expenditure, labor cost, and software development expenditure. Several developers and researchers [15,82] show their concerns over the cost issue macroscopically. CSPs [1−4] release their FPGA-equipped computing instances based on the considerations over the costs of FPGA deployment.

Cloud FPGA systems should be installed with particular supports to meet efficiency and security needs. The deployment of hardware supports, software supports, and security guarantees account for the majority of the preparatory expenditures. The following is a breakdown of the costs.

Facility expenditure These costs are mainly used for the hardware deployment. CSPs [1−5,15,82] show concerns of this region, and release their FPGA instances considering the facility costs. These costs can be quantified by the amount of the facility purchase money.

Prevention costs These security costs contain encryption costs [73,74] and function isolation costs [3]. Security guarantee in preparation are mainly used to separate the hardware design of tenants. These hidden costs appear as encryption and functional isolation overheads, which may be measured as resource occupancy and development fees.

Labor costs FPGA sharing necessitates software support [14,18,30,31], which would cost money in terms of engineering. Management, virtualization, and communication expenses are all affected by FPGA sharing.

5.2 Execution costs

Execution expenses are incurred during the execution process. During run-time, these costs are represented as overheads. The criteria for FPGA sharing hguanzhave an influence on execution costs. Execution costs are rarely immediately quantifiable in terms of money. These overheads come from

Table 5 Costs of FPGA sharing, categorised by the execution stage, cost type and quantification

Stage	Cost types	Source	Explanation	Quantity	Related work
Preparation	Prep. Costs	Hardware	Expenditure of Facility	Money Cost ($)	[1–5,15,82]
		Security	Prevention Costs: Encryption/Isolation	Overhead (**O** & $)	[3,73,74]
		Software	Development Labor Costs	Money Cost ($)	[14,18,30,31]
Execution	Exec. Costs	Security	Power Control/Assets Protection	Overhead (**O** & **D**)	[76,77,86]
		Software	Scheduling/Data/Execution	Overhead (**D**)	[21,49,88,89]
Release	Post Costs	Security	Configuration Memory Refresh	Overhead (**D**)	[11,80,90]

* O represents the occupation, D represents the duration, and $ represents Dollars.

the consumption or the occupation of resources in processing. Several works present their concern towards these execution related costs. Mocha [88] provides a sensible cost model to illustrate the issue of cost-efficiency of cloud FPGA computing. Shepovalov et al. [89] execute tasks in the cloud and provide their thinking over the economic computing. There are also some works concerning the overheads of the virtualization of FPGA. QMC [83] provides a method to quantify the cost of virtualization. Vital [18] provides the comparison between several virtualization overheads of existing techniques. Execution expenses are mostly derived from two sources.

Maintenance costs To guarantee the service quality in multi-tenant scenarios, FPGA sharing security measures need to be taken into account [11,13,73,80]. The countermeasures for severe threats will increase the burden for maintenance. Some security are directly base on the encryption [10,74,87]. Several safety defenses of treats rely on consuming more resources to the area. There are also other security guarantees require specific SW/HW supports [3,90], which improves the costs for supporting deployment.

Management overheads Fine control increases the development costs in software [15,46]. CSPs need extra expenditure for developing the supporting system. These systems are required for providing the appropriate abstract and resource management. Importantly, specific spatial sharing calls for the effective management of fine-grained partitions [18,21,30,45,46,56]. The supporting development will add up maintenance costs. The temporal sharing [14,56] needs to properly allocate the contention resources. FPGA-instances price will be influenced by the management based on the consideration. The sharing of FPGA sharing will lead to higher network pressure inner hardware.

5.3 Post cost

Post expenses are considered for asset protection in the release stage,. Computing jobs complete the processing with results at this stage. For function updates, the configuration of co-locating designs would be evicted. Design protection is a serious challenge to FPGA sharing in multi-tenancy. With side-channeling, the evil tenants or suppliers would steal the design assets [10,13]. To avoid design leaks at the release stage, post protection is required.

Post-protection costs The release's FPGA sharing expenses are influenced by security needs. In monetary terms, post-expenses are rarely readily quantified. Post expenses are rarely immediately quantifiable in monetary terms. Configuration memory refresh is the major source of post-processing expenses. One apparent way to prevent design assets from leaking is to update configuration memory [11,80,90]. The duration may be used to calculate these overheads.

6 Challenges and opportunities

Although many prior technologies have been studied for the optimization of FPGA sharing in different aspects, several challenges remain open. In this section, we identify the following challenges and opportunities for developing FPGA sharing in the cloud with high cost-efficiency.

New FPGA configuration mechanism It is crucial to explore a new FPGA configuration mechanism to mitigate the huge overhead. The future deployment of FPGAs in cloud can be a hybrid model. The configuration switches when FPGAs are temporally shared by different applications is unavoidable time-consuming. Developers can configure FPGA according to the characterization of sharing applications. For small-scale tasks, they consume little on-chip area, which can configured as a whole in FPGA proactively. While for large-scale tasks, developers need to partially configure FPGAs to support the common components of sharing applications with other components remaining un-configured. The pre-configured FPGA can reduce the configuration overhead.

Fine-grained resource sharing Fine-grained resource sharing is the key to FPGA sharing. The fine-grained resource sharing not only requires appropriate hardware abstraction, but also needs to handle resources sharing in real-time system. In this case, the current FPGA virtualization is not enough to support fine-grained resource sharing in multi-tenancy scenarios which will cause communication blocking, clock blocking and other issues.

In addition to FPGA virtualization, another opportunity for fine-grained resource sharing is kernel-level FPGA hardware development. The FPGA kernel is the code form of the hardware circuit running on the FPGA board. With kernel integration and kernel fusion, one can manage the FPGA resources in the kernel level and save resources. In this case, the FPGA sharing system should be able to identify the characterization of applications and fuse their kernel without performance degradation.

Compilation-based FPGA sharing Compilation is an important step in FPGA developing, especially in FPGA sharing. Leveraging the compilation optimization can enhance the performance and efficiency of FPGA sharing. Different from single application compilation in FPGA, the compilation techniques in FPGA sharing should consider the optimization space of multi application which shares the same FPGA. By adding another layer of abstraction, the compiler can provide isolation without sacrificing performance. The multiple FPGA "roles" do not have to be physically isolated from each other, thus making placement of LUTs and wiring much easier [92]. In addition to the optimization mechanism of compilation, optimizing the compilation tools and compilation performance is also a non-trivial problem.

Security and resiliency The security guarantee is a common challenge in multi-tenancy scenarios. Especially for FPGA, the sharing granularity is finer than other hardware and platforms. The open hardware configuration process of FPGA makes the hardware a hidden trouble in cloud sharing scenarios. An efficient FPGA sharing require the resiliency, and a mechanism should cover the security problems including bypass attacks, design leaks, power attacks and others.

Cost efficiency Prior art of the FPGA sharing usually focuses on the feasibility of implementation. The optimization of resource utilization is an open research region. Developers should design new optimization mechanisms to improve the utilization of FPGAs using FPGA sharing. Expect for

utilization improvement, performance-per-dollar is also a high-profile interest of cloud computing. The costs of FPGA sharing have various sources. Correspondingly, there are many opportunities for optimizing the cost-efficiency of cloud FPGA in multi-tenants scenarios. However, very limited works focus on the optimization over cost problems.

7 Conclusion

In this work, we examine FPGA implementation in the cloud and summarized the characteristics of FPGA sharing. In multi-tenancy scenarios, FPGA sharing plays an important role. We discuss the related works of FPGA sharing based on the organization of resources. FPGA sharing has flexibility in manners and granularity. We also provide a detailed analysis of the SW/HW sharing support for sharing and discuss design cost. Finally, combined with our observations, we present the key challenges and opportunities of FPGA sharing.

Acknowledgements We thank all the reviewers for their valuable comments and feedbacks. This work was sponsored by the National Natural Science Foundation of China (Grant No. 61972247).

References

1. Amazon Cloud. Amazon EC2 F1 Instances. See Aws.amazon.com/cn/ec2/instance-types/f1/ website, 2016
2. Huawei Cloud. FPGA Accelerated Cloud Server. See Huaweicloud.com/product/facs.html website, 2017
3. Alibaba Cloud. FPGA Cloud Server. See Aliyun.com/product/ecs/fpga website, 2017
4. Baidu AI Cloud. FPGA Cloud Server. See Cloud.baidu.com/product/fpga.html website, 2017
5. Tencent Cloud. FPGA Cloud Server. See Cloud.tencent.com/product/fpga website, 2018
6. Tourad E H C, Eleuldj M. Survey of deep learning neural networks implementation on FPGAs. In: Proceedings of the 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications. 2020, 1−8
7. Wu C, Fresse V, Suffran B, Konik H. Accelerating DNNs from local to virtualized FPGA in the cloud: a survey of trends. Journal of Systems Architecture, 2021, 119: 102257
8. Shahzad H, Sanaullah A, Herbordt M C. Survey and future trends for FPGA cloud architectures. In: Proceedings of 2021 IEEE High Performance Extreme Computing Conference. 2021, 1−10
9. Quraishi M H, Tavakoli E B, Ren F. A survey of system architectures and techniques for FPGA virtualization. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(9): 2216–2230
10. Trimberger S, McNeil S. Security of FPGAs in data centers. In: Proceedings of IEEE the 2nd International Verification and Security Workshop. 2017, 117−122
11. Turan F, Verbauwhede I. Trust in FPGA-accelerated cloud computing. ACM Computing Surveys, 2021, 53(6): 128
12. Jin C, Gohil V, Karri R, Rajendran J. Security of cloud FPGAs: a survey. 2020, arXiv preprint arXiv: 2005.04867
13. Dessouky G, Sadeghi A R, Zeitouni S. SoK: Secure FPGA multi-tenancy in the cloud: challenges and opportunities. In: Proceedings of 2021 IEEE European Symposium on Security and Privacy. 2021, 487−506
14. Chen F, Shan Y, Zhang Y, Wang Y, Franke H, Chang X, Wang K. Enabling FPGAs in the cloud. In: Proceedings of the 11th ACM Conference on Computing Frontiers. 2014, 3
15. Putnam A, Caulfield A M, Chung E S, Chiou D, Constantinides K, Demme J, Esmaeilzadeh H, Fowers J, Gopal G P, Gray J, Haselman M, Hauck S, Heil S, Hormati A, Kim J Y, Lanka S, Larus J, Peterson E, Pope S, Smith A, Thong J, Xiao P Y, Burger D. A reconfigurable fabric for accelerating large-scale datacenter services. In: Proceedings of 2014 ACM/IEEE 41st International Symposium on Computer Architecture. 2014, 13−24
16. Intel. Hardware accelerator research program. See Communityintel.com/t5/Software-Tuning-Performance/HARP-Hardware-Accelerator-Research-Program/m-p/1126570 website, 2018
17. Iordache A, Pierre G, Sanders P, De F. Coutinho J G, Stillwell M. High performance in the cloud with FPGA groups. In: Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing. 2016, 1−10
18. Zha Y, Li J. Virtualizing FPGAs in the cloud. In: Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. 2020, 845−858
19. Byma S, Steffan J G, Bannazadeh H, Leon-Garcia A, Chow P. FPGAS in the cloud: booting virtualized hardware accelerators with OpenStack. In: Proceedings of the 22nd IEEE Annual International Symposium on Field-Programmable Custom Computing Machines. 2014, 109−116
20. Lavin C, Kaviani A. RapidWright: enabling custom crafted implementations for FPGAs. In: Proceedings of the 26th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines. 2018, 133−140
21. Zhu Z, Liu A X, Zhang F, Chen F. FPGA resource pooling in cloud computing. IEEE Transactions on Cloud Computing, 2021, 9(2): 610–626
22. Guo L, Chi Y, Wang J, Lau J, Qiao W, Ustun E, Zhang Z, Cong J. AutoBridge: coupling coarse-grained floorplanning and pipelining for high-frequency HLS design on multi-die FPGAs. In: Proceedings of the 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2021, 81−92
23. Moghaddamfar M, Färber C, Lehner W, May N, Kumar A. Resource-efficient database query processing on FPGAs. In: Proceedings of the 17th International Workshop on Data Management on New Hardware. 2021, 4
24. Chen X, Tan H, Chen Y, He B, Wong W F, Chen D. ThunderGP: HLS-based graph processing framework on FPGAs. In: Proceedings of 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2021, 69−80
25. Liu C, Shao Z, Li K, Wu M, Chen J, Li R, Liao X, Jin H. ScalaBFS: a scalable BFS accelerator on FPGA-HBM platform. In: Proceedings of 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2021, 147
26. Wu Y W, Wang Q G, Zheng L, Liao X F, Jin H, Jiang W B, Zheng R, Hu K. FDGLib: a communication library for efficient large-scale graph processing in FPGA-accelerated data centers. Journal of Computer Science and Technology, 2021, 36(5): 1051–1070
27. Bacis M, Brondolin R, Santambrogio M D. BlastFunction: an FPGA-as-a-service system for accelerated serverless computing. In: Proceedings of the 23rd Conference on Design, Automation and Test in Europe. 2020, 852−857
28. Hong B, Kim H Y, Kim M, Suh T, Xu L, Shi W. FASTEN: an FPGA-based secure system for big data processing. IEEE Design & Test, 2018, 35(1): 30–38
29. Skhiri R, Fresse V, Malek J, Suffran B, Jamont J. An approach for an efficient sharing of IP as a service in cloud FPGA. In: Proceedings of

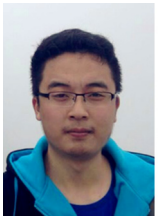
the 18th International Multi-Conference on Systems, Signals & Devices. 2021, 784−789

30. Khawaja A, Landgraf J, Prakash R, Wei M, Schkufza E, Rossbach C J. Sharing, protection, and compatibility for reconfigurable fabric with Amorphos. In: Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation. 2018, 107−127
31. Zha Y, Li J. Hetero-ViTAL: A virtualization stack for heterogeneous FPGA clusters. In: Proceedings of the 48th ACM/IEEE Annual International Symposium on Computer Architecture. 2021, 470−483
32. Xilinx. Alveo data center accelerator card platforms. See Docs.xilinx.com/r/en-US/ug1120-alveo-platforms website, 2021
33. Intel. FPGA Cloud Server, High Bandwidth Memory (HBM2) Interface Intel® FPGA IP user guide. See Intel website, 2021
34. Choi Y K, Cong J, Fang Z, Hao Y, Reinman G, Wei P. In-depth analysis on microarchitectures of modern heterogeneous CPU-FPGA platforms. ACM Transactions on Reconfigurable Technology and Systems, 2019, 12(1): 4
35. Voss N, Quintana P, Mencer O, Luk W, Gaydadjiev G. Memory mapping for multi-die FPGAs. In: Proceedings of the 27th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines. 2019, 78−86
36. Xilinx. Alveo U280 data center accelerator card user guide. See Xilinx.com/products/boards-and-kits/alveo/u280 website, 2021
37. Hung J R, Li C, Wang P, Shao C, Guo J, Wang J, Shi G. ACE-GCN: a fast data-driven FPGA accelerator for GCN embedding. ACM Transactions on Reconfigurable Technology and Systems, 2021, 14(4): 21
38. Xilinx. Xilinx UltraScale: The next-generation architecture for your next-generation architecture. See Docs.xilinx.com/v/u/en-US/wp435-Xilinx-UltraScale website. 2014
39. Ma J, Zuo G, Loughlin K, Cheng X, Liu Y, Eneyew A M, Qi Z, Kasikci B. A hypervisor for shared-memory FPGA platforms. In: Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. 2020, 827−48444
40. Tarafdar N, Eskandari N, Lin T, Chow P. Designing for FPGAs in the cloud. IEEE Design & Test, 2018, 35(1): 23–29
41. Vaishnav A, Pham K D, Koch D, Garside J. Resource elastic virtualization for FPGAs using OpenCL. In: Proceedings of the 28th International Conference on Field Programmable Logic and Applications. 2018, 111−41184
42. Gonzalez I, Lopez-Buedo S, Sutter G, Sanchez-Roman D, Gomez-Arribas F J, Aracil J. Virtualization of reconfigurable coprocessors in HPRC systems with multicore architecture. Journal of Systems Architecture, 2012, 58(6−7): 247−256
43. Huang C H, Hsiung P A. Virtualizable hardware/software design infrastructure for dynamically partially reconfigurable systems. ACM Transactions on Reconfigurable Technology and Systems, 2013, 6(2): 11
44. Wang W, Bolic M, Parri J. pvFPGA: accessing an FPGA-based hardware accelerator in a paravirtualized environment. In: Proceedings of 2013 International Conference on Hardware/Software Codesign and System Synthesis. 2013, 10
45. Kidane H L, Bourennane E B, Ochoa-Ruiz G. NoC based virtualized accelerators for cloud computing. In: Proceedings of the 10th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip. 2016, 133−137
46. Vatsolakis C, Pnevmatikatos D. RACOS: transparent access and virtualization of reconfigurable hardware accelerators. In: Proceedings of 2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. 2017, 11−19
47. Zhao Q, Amagasaki M, Iida M, Kuga M, Sueyoshi T. Enabling FPGA-as-a-service in the cloud with hCODE platform. IEICE Transactions on Information and Systems, 2018, 101-D(2): 335–343
48. Mbongue J M, Hategekimana F, Kwadjo D T, Bobda C. FPGA virtualization in cloud-based infrastructures over virtio. In: Proceedings of the 36th IEEE International Conference on Computer Design. 2018, 242−245
49. Mbongue J, Hategekimana F, Kwadjo D T, Andrews D, Bobda C. FPGAVirt: a novel virtualization framework for FPGAs in the cloud. In: Proceedings of the 11th IEEE International Conference on Cloud Computing. 2018, 862−865
50. Coole J, Stitt G. Intermediate fabrics: virtual architectures for circuit portability and fast placement and routing. In: Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis. 2010, 13−22
51. Cong J, Wei P, Yu C H, Zhou P. Bandwidth optimization through on-chip memory restructuring for HLS. In: Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference. 2017, 43
52. Yang J, Yan L, Ju L, Wen Y, Zhang S, Chen T. Homogeneous NoC-based FPGA: the foundation for virtual FPGA. In: Proceedings of the 10th IEEE International Conference on Computer and Information Technology. 2010, 62−67
53. Huang C, Hsiung P A. Hardware resource virtualization for dynamically partially reconfigurable systems. IEEE Embedded Systems Letters, 2009, 1(1): 19–23
54. Asiatici M, George N, Vipin K, Fahmy S A, Ienne P. Designing a virtual runtime for FPGA accelerators in the cloud. In: Proceedings of the 26th International Conference on Field Programmable Logic and Applications. 2016, 1−2
55. Giechaskiel I, Rasmussen K, Szefer J. Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs. In: Proceedings of the 37th IEEE International Conference on Computer Design. 2019, 1−10
56. Knodel O, Lehmann P, Spallek R G. RC3E: reconfigurable accelerators in data centres and their provision by adapted service models. In: Proceedings of the 9th IEEE International Conference on Cloud Computing. 2016, 19−26
57. Huang M, Wu D, Yu C H, Fang Z, Interlandi M, Condie T, Cong J. Programming and runtime support to blaze FPGA accelerator deployment at datacenter scale. In: Proceedings of the 7th ACM Symposium on Cloud Computing. 2016, 456−469
58. Stangl J, Lor¨unser T, Dinakarrao S M P. A fast and resource efficient FPGA implementation of secret sharing for storage applications. In: Madsen J, Coskun A K, eds, 2018 Design, Automation & Test in Europe Conference & Exhibition. 2018, 654–659
59. Weerasinghe J, Abel F, Hagleitner C, Herkersdorf A. Enabling FPGAs in hyperscale data centers. In: Proceedings of 2015 IEEE the 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE the 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE the 15th Intl Conf on Scalable Computing and Communications and its Associated Workshops. 2015, 1078−1086
60. Fahmy S A, Vipin K, Shreejith S. Virtualized FPGA accelerators for efficient cloud computing. In: Proceedings of the 7th IEEE International Conference on Cloud Computing Technology and Science. 2015, 430−435
61. Li X, Wang X, Liu F, Xu H. DHL: enabling flexible software network functions with FPGA acceleration. In: Proceedings of the 38th IEEE International Conference on Distributed Computing Systems. 2018,

1−11

62. Papadimitriou K, Dollas A, Hauck S. Performance of partial reconfiguration in FPGA systems: a survey and a cost model. ACM Transactions on Reconfigurable Technology and Systems, 2011, 4(4): 36
63. Xilinx. UltraScale Architecture Configuration. See Docs.xilinx.com/v/u/en-US/ug570-ultrascale-configuration website, 2022
64. Zhang J, Xiong Y, Xu N, Shu R, Li B, Cheng P, Chen G, Moscibroda T. The feniks FPGA operating system for cloud computing. In: Proceedings of the 8th Asia-Pacific Workshop on Systems. 2017, 22
65. Knodel O, Genssler P R, Spallek R G. Migration of long-running tasks between reconfigurable resources using virtualization. ACM SIGARCH Computer Architecture News, 2016, 44(4): 56–61
66. Vaishnav A, Pham K, Koch D. Live migration for OpenCL FPGA accelerators. In: Proceedings of 2018 International Conference on Field-Programmable Technology. 2018, 38−45
67. Korolija D, Roscoe T, Alonso G. Do OS abstractions make sense on FPGAs? In: Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation. 2020, 991−1010
68. Asiatici M, George N, Vipin K, Fahmy S A, Ienne P. Virtualized execution runtime for FPGA accelerators in the cloud. IEEE Access, 2017, 5: 1900–1910
69. Fowers J, Ovtcharov K, Papamichael M, Massengill T, Liu M, Lo D, Alkalay S, Haselman M, Adams L, Ghandi M, Heil S, Patel P, Sapek A, Weisz G, Woods L, Lanka S, Reinhardt S K, Caulfield A M, Chung E S, Burger D. A configurable cloud-scale DNN processor for real-time AI. In: Proceedings of the 45th ACM/IEEE Annual International Symposium on Computer Architecture. 2018, 1−14
70. Yazdanshenas S, Betz V. Improving confidentiality in virtualized FPGAs. In: Proceedings of 2018 International Conference on Field-Programmable Technology. 2018, 258−261
71. Yazdanshenas S, Betz V. The costs of confidentiality in virtualized FPGAs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019, 27(10): 2272–2283
72. Zhao M, Gao M, Kozyrakis C. ShEF: shielded enclaves for cloud FPGAs. In: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022, 1070−1085
73. Krautter J, Gnad D R E, Tahoori M B. Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud. ACM Transactions on Reconfigurable Technology and Systems, 2019, 12(3): 12
74. Krautter J, Gnad D, Tahoori M. CPAmap: On the complexity of secure FPGA virtualization, multi-tenancy, and physical design. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020, 2020(3): 121–146
75. Duncan A, Nahiyan A, Rahman F, Skipper G, Swany M, Lukefahr A, Farahmandi F, Tehranipoor M. SeRFI: Secure remote FPGA initialization in an untrusted environment. In: Proceedings of the 38th IEEE VLSI Test Symposium. 2020, 1−6
76. Krautter J, Gnad D R E, Schellenberg F, Moradi A, Tahoori M B. Active fences against voltage-based side channels in multi-tenant FPGAs. In: Proceedings of 2019 IEEE/ACM International Conference on Computer-Aided Design. 2019, 1−8
77. Jayasinghe D, Ignjatovic A, Parameswaran S. RFTC: runtime frequency tuning countermeasure using FPGA dynamic reconfiguration to mitigate power analysis attacks. In: Proceedings of the 56th ACM/IEEE Design Automation Conference. 2019, 1−6
78. Provelengios G, Holcomb D, Tessier R. Characterizing power distribution attacks in multi-user FPGA environments. In: Proceedings of the 29th International Conference on Field Programmable Logic and Applications. 2019, 194−201
79. Walder H, Platzner M. A runtime environment for reconfigurable hardware operating systems. In: Proceedings of the 14th International Conference on Field Programmable Logic and Application. 2004, 831−835
80. Ko H J, Li Z, Midkiff S. Optimizing data layout and system configuration on FPGA-based heterogeneous platforms. In: Proceedings of 2018 IEEE/ACM International Conference on Computer-Aided Design. 2018, 1−8
81. Koch D. Partial Reconfiguration on FPGAs: Architectures, Tools and Applications. New York: Springer, 2013
82. Caulfield A M, Chung E S, Putnam A, Angepat H, Fowers J, Haselman M, Heil S, Humphrey M, Kaur P, Kim J Y, Lo D, Massengill T, Ovtcharov K, Papamichael M, Woods L, Lanka S, Chiou D, Burger D. A cloud-scale acceleration architecture. In: Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture. 2016, 1−13
83. Yazdanshenas S, Betz V. Quantifying and mitigating the costs of FPGA virtualization. In: Proceedings of the 27th International Conference on Field Programmable Logic and Applications. 2017, 1−7
84. Kirchgessner R, Stitt G, George A, Lam H. VirtualRC: a virtual FPGA platform for applications and tools portability. In: Proceedings of the 20th ACM/SIGDA International Symposium on Field Programmable Gate Arrays. 2012, 205−208
85. Stitt G, Karam R, Yang K, Bhunia S. A uniquified virtualization approach to hardware security. IEEE Embedded Systems Letters, 2017, 9(3): 53–56
86. Schellenberg F, Gnad D R E, Moradi A, Tahoori M B. An inside job: remote power analysis attacks on FPGAs. In: Proceedings of 2018 Design, Automation & Test in Europe Conference & Exhibition. 2018, 1111−1116
87. Trimberger S. Security in SRAM FPGAs. IEEE Design & Test of Computers, 2007, 24(6): 581
88. Zhou P, Sheng J, Yu C H, Wei P, Wang J, Wu D, Cong J. MOCHA: multinode cost optimization in heterogeneous clouds with accelerators. In: Proceedings of 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2021, 273−279
89. Shepovalov M, Akella V. FPGA and GPU-based acceleration of ML workloads on amazon cloud - A case study using gradient boosted decision tree library. Integration, 2020, 70: 1–9
90. Eguro K, Venkatesan R. FPGAs for trusted cloud computing. In: Proceedings of the 22nd International Conference on Field Programmable Logic and Applications. 2012, 63−70
91. Wu C, Buyya R, Ramamohanarao K. Cloud pricing models: taxonomy, survey, and interdisciplinary challenges. ACM Computing Surveys, 2020, 52(6): 108
92. Landgraf J, Yang T, Lin W, Rossbach C J, Schkufza E. Compiler-driven FPGA virtualization with SYNERGY. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2021, 818−831

Jinyang Guo received the bachelor degree in software engineering from Wuhan University, China. He is working toward the master degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His main research interests include graph processing, FPGA accelerator, and edge computing.



Lu Zhang received the BS degree from the Northwestern Polytechnical University, China in 2016. He is working toward the PhD degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research interests include edge computing, network function virtualization, and serverless computing.

José Romero Hung is an International PhD Candidate in the Department of Micro/Nano Electronics, Shanghai Jiao Tong University, China. His most recent research contributions include topics on hardware acceleration for complex machine learning applications.

Chao Li received his PhD degree from the University of Florida, USA in 2014. He is currently an professor with tenure in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His research mainly focuses on computer architecture and systems for emerging applications. He is a senior member of IEEE/ACM/CCF.

Jieru Zhao received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, China in 2020. She is an Assistant Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. Her current research interests include reconfigurable system, high-level synthesis and HW/SW co-design for emerging applications. Dr. Zhao was a recipient of the Best Paper Award at ICCAD 2017 and Best Paper Nominations at DATE 2022, SC 2021, and CASES 2018.

Minyi Guo received his PhD degree in computer science from the University of Tsukuba, Japan. He is currently Zhiyuan Chair professor, Shanghai Jiao Tong University, China. His research interests include parallel/distributed computing, compiler optimizations, cloud computing, and big data. He is now on the editorial board of IEEE Transactions on Parallel and Distributed Systems and Journal of Parallel and Distributed Computing. Dr. Guo is IEEE fellow and CCF fellow.