Hindawi
International Journal of Reconfigurable Computing
Volume 2019, Article ID 8085461, 17 pages
https://doi.org/10.1155/2019/8085461

Review Article

From FPGA to Support Cloud to Cloud of FPGA: State of the Art

Rym Skhiri,[1] **Virginie Fresse**,[2] **Jean Paul Jamont**,[3] **Benoit Suffran**,[4] **and Jihene Malek**[1,5]

[1]*Electronics and Microelectronics Lab, LR99ES30, Faculty of Sciences of Monastir, University of Monastir, Monastir 5000, Tunisia*
[2]*Hubert Curien Lab, UMR CNRS 5516, Jean Monnet University, University of Lyon, Saint-Etienne 42000, France*
[3]*LCIS Laboratory, Univ. Grenoble Alpes, Valence 26000, France*
[4]*STMicroelectronics, Grenoble 38000, France*
[5]*Higher Institute of Applied Sciences and Technology of Sousse, Sousse University, Sousse 4003, Tunisia*

Correspondence should be addressed to Virginie Fresse; virginie.fresse@univ-st-etienne.fr

Received 13 June 2019; Revised 23 September 2019; Accepted 6 November 2019; Published 5 December 2019

Academic Editor: Martin Margala

Field Programmable Gate Array (FPGA) draws a significant attention from both industry and academia by accelerating computationally expensive applications and achieving low power consumption. FPGAs are interesting due to the flexibility and reconfigurabiltiy of their device. Cloud computing becomes a major trend towards infrastructure and computing resources dematerialization. It provides "unlimited" storage capacities and a large number of data and applications that make collaboration easier between multiple (not domain specific) designers. Many papers in the literature have surveyed Cloud and FPGA separately and, more precisely, their services and challenges. The acceleration of applications by FPGA and the unlimited capacities of the cloud are expected to be more and more pervasive. As more and more FPGA are being deployed in traditional cloud, it is appropriate to clarify what is the cloud FPGA and which drawbacks of using FPGA in local are resolved. We present a survey of the cloud FPGA works that have been proposed to exploit the advantages of using FPGA in the cloud. We classify these studies in three services to highlight their benefits and limitations. This survey aims at motivating further researches in cloud FPGA.

1. Introduction

Field Programmable Gate Array (FPGA) is an integrated reconfigurable device, composed of reprogrammable logic blocks. These logic blocks are connected through a configurable interconnection and input/output blocks to execute a specific application. FPGA ensures the reconfiguration of its architecture for each new application. FPGAs are highly used for compute intensive applications due their efficient power consumptions and their fast execution times. For instance, in the field of signal processing, the sliding-window application [1] shows that the FPGA offers better energy efficiency and can achieve a speed up of up to 11x and 57x compared to GPU and multicores. Furthermore, FPGA can be a better choice for some types of image processing applications, for example, the stereo vision application [2]. However, designing an FPGA architecture takes a long time, and the associated traditional design flow requires hardware skills such as Hardware Description Language (HDL) and hardware tools [3].

The cloud computing is a paradigm that is defined by the National Institute of Standards and Technology (NIST) as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, and applications) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [4, 5]. The cloud computing provides three services, respectively: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These services can be exploited anywhere through internet by a pay-per-use model [6]. These resources can be dynamically scalable making possible to meet the performance objective required by applications. Nevertheless, the cloud involves various issues to be resolved [7–9]: (i) the increase of power over a cluster of servers, (ii)

the varying response times, (iii) the security and privacy of data, and (iv) the standardizing of cloud technology. FPGA can respond of the answer.

In this paper, we present a survey of cloud FPGA. Our aim is to provide a better understanding of the current research issues in this emerging field. Several states of the art present cloud computing applications that have implemented FPGA as common hardware in data centers to accelerate remote applications [10–12]. Some surveys focus on the virtualization of FPGA in cloud [13–15] to provide an abstraction of the used FPGA hardware. Other states of the art focus on using FPGA to enhance cloud system security [6, 16, 17]. Table 1 presents the existing surveys in the literature and our survey according to two axes. The first axis focuses on the existing surveys on how FPGAs supports the infrastructure of the cloud. In this case, FPGAs are used to speed up the application and to reduce energy consumption in data center infrastructure. These infrastructures contain CPUs and/or GPUs and FPGA. In this context, FPGA can also be used in the infrastructure to secure cloud services and user data. The second axis presents the existing solutions of cloud of FPGAs. In this case, only FPGA are in the cloud as computing resources. In this context, several services are proposed for such FPGA cloud. Three services are proposed and only the Resources as a service (RaaS) is considered in some surveys of cloud of FPGAs. From what we know, there is no survey about Platform as a service (PaaS) or Software as a service (SaaS). The objective of this paper is to present the main challenges of FPGA design in local, a survey of existing services for cloud of FPGAs, with the challenges solved with these services. The survey of these services in the cloud of FPGA will point out these services tackle challenges of FPGA design and validation.

The idea is to help application designers to execute their FPGA applications in the cloud using remote FPGA software tools or remote FPGA platforms or remote FPGA resources. The cloud integrates FPGA to benefit from their resources to support their cloud infrastructures. The multiple application designers can efficiently use the FPGA resources by taking advantage of the "unlimited" capabilities of the cloud. Towards the end goal of a thorough comprehension of the relationship between cloud computing and FPGA, we will discuss the different facets of cloud FPGA from using FPGA in the cloud as "an acceleration as a service" to consider the cloud as a support on executing FPGA applications. The cloud FPGA can refer here to the use of multiple FPGAs, multiple of their resources to form a single system. A common use of the cloud FPGA is to provide multiple application designers to design and implement their own applications from a remote location in the FPGA platform.

The paper is therefore organized as follow. In Section 2, we briefly cover a background of the cloud computing by providing the necessary basics and cloud services needed to tackle the original concept of the cloud FPGA. Based on the advantages of FPGA, we then outline the different approaches of integrating FPGA in cloud infrastructure to significantly speed up many intensive tasks and achieve a better energy. Section 3 presents the drawbacks of designing and using an FPGA by one designer in local. Based on the cloud computing services, we have classified the services of cloud FPGA in three main categories. In Section 4, we detail a literature review and a discussion for each service. The first service models introduce the different solutions related to enable the migration of FPGA tools in the cloud. The second service model presents some works related to accessing and using FPGA boards remotely. In the third model, we present works that are interested in sharing FPGA resources between multiple application designers. Before concluding, we summarize general discussions and present some future directions in Section 5.

2. Basic Concepts of Cloud Computing

The cloud computing is a model for enabling ubiquitous, on-demand access to a shared pool of scalable physical resources such as physical servers and interconnection elements (routers, switches, etc) resources. We introduce firstly some definitions and the background of cloud computing. Then, we describe the reliable services delivered by the cloud computing through data centers.

2.1. Definitions. The cloud supports a shift of computing and storage resources from local to the network. The cloud is distributed from cloud vendors to cloud users through virtual data centers. A data center is a large group of networked resources typically used for the remote storage and computation of large amount of resources [18]. The cloud vendor delivers various types of services that cloud users can access. The cloud provides several service models based on the virtualization and dematerialization techniques. Virtualization aims at splitting software and hardware resources into different parts, and each part operates in its own independent manner and runs on Virtual Machine. Dematerialization [19] aims at moving hardware and software resources to be remotely used.

2.2. Benefits of Cloud Computing. From the definition provided by the NIST, the idea behind cloud computing is based on a set of features that are different from traditional service computing:

(i) Ubiquitous access: clouds are generally accessible through the Internet [20]. Any device with network connectivity allows users to access cloud services by heterogeneous client platforms such as smart phones, laptops, and workstation computers.

(ii) On-demand: the cloud computing is a web-based processing, by which shared resources and applications are provided on demand. This allows cloud vendors and users to adjust their computing capacity depending on a given task at a given time.

(iii) Multitenancy: multiple cloud vendors offer their own services in a single data center [21]. The resources are pooled to serve multiuser using a multitenant model in that the cloud user does not have any control or knowledge over the exact location. This means that the cloud does not care who

Table 1: Overview of several surveys introducing FPGA in cloud.

Related works	FPGA to support cloud			Cloud of FPGA		
	Acceleration	Energy reduction	Security of the cloud	SaaS	PaaS	RaaS
Kachris et al. [10]	✓	✓				
Lee et al. [11]	✓	✓				
Mohammedali et al. [12]	✓	✓				
Vaishnav [13]						✓
Le et al. [14]						✓
Vipin et al. [15]						✓
Mondol et al. [16]			✓			
Will et al. [17]			✓			
This survey				✓	✓	✓

is using which processor or memory. Multicloud users may be using the same resources at the same time [22].

(iv) Elasticity and scalability: traditional computing services provide a fixed number of resources in a fixed amount of time. In cloud computing, cloud users can instantly scale up or down computing resources according to their own needs. Capabilities can be elastically provisioned and released, sometimes automatically, to scale rapidly outward and inward commensurate with demand [23]. When they are not needed anymore, resources can be scaled back to their original states.

(v) Measured usage: cloud computing employs a pay-per-use pricing model, which differs from service to service. The measured usage is the ability to keep track of the resources usage of users. A vendor may rent a virtual machine from another vendor for delivering its users [24]. The consumption of computing resources can be measured by calculating the usage that users actually consume.

2.3. Cloud Computing Services. The cloud computing offers to cloud users three types of services using virtualization and dematerialization techniques. In each service, the user has different levels of control and management in order to choose the suitable service for his needs.

(i) SaaS: this model allows users to access and uses only cloud applications which are software programs (often available via web browsers). This is an alternative method to locally run applications. The user does not need to invest expensive costs in license payments and updates. The SaaS reduces implementation and maintenance costs [25, 26]. The cloud vendor controls and manages the application level [27].

(ii) PaaS: this model presents the middle bridge between hardware and software because it abstracts the infrastructure and supports a set of applications that can be run on the cloud platform. Users can develop and run their applications using programming languages, libraries, and tools without needing to buy the software and hardware resources [28]. Users can benefit from the elasticity and scalability of cloud applications. In this way, users have the possibility to adjust their computational resources (e.g., memory and storage disk) according to their needs.

(iii) IaaS: this model enables to rent an infrastructure which is a collection of servers, storage resources, and operating systems that are needed to build applications [29]. The cloud vendor manages the cloud infrastructure and the user can deploy their applications on these rented infrastructures [22]. In IaaS, the virtualization technique is used to share resources between users.

2.4. FPGA for Cloud Infrastructures. FPGA starts to appear recently in commercial cloud platforms such as Microsoft [29], Amazon [30], and others to respond to some cloud issues (improve power over a cluster of servers and the varying response times). The data center deploys FPGA in their infrastructures with two main approaches, either FPGA tightly coupled to the Central Processing Unit (CPU) or FPGA as a standalone component.

The first approach considers FPGA as a coprocessor. FPGA and CPU are physically connected together, and the CPU is also connected to the network. In this case, FPGA becomes both an accelerator and a part of the data center. However, the number of FPGA in the data center is limited to the number of CPU, and FPGA cannot be used as an independent computing resource. In this approach, Amazon [31] has delivered the F1 instances type which integrates Xilinx Virtex UltraScale+. Amazon uses FPGA instances to accelerate some tasks ranging from analytics and machine learning to databases and network virtualization. The Amazon F1 instance achieves 10x better cost efficiency than Amazon's CPU Elastic Cache [32]. The CAPI solution from IBM is similar to that from Amazon; it deploys the Xeon processor with FPGA in the same package [33]. Microsoft released the Catapult project in 2014 [34, 35] which deployed one Altera Stratix vFPGA per CPU. Microsoft connects FPGAs directly to the network but does not make them directly accessible and programmable by hardware designers. Catapult is difficult to be adapted to different applications. The target application accelerates the Bing web search engine with achieving an improvement of 95% in

throughput improvement while consuming only 10% more power per CPU-FPGA server. These tightly coupled servers enable an acceleration of local applications to meet performance demands.

The second approach considers FPGA as a standalone disaggregated component [36] independent from the CPU. FPGA is then directly connected to the network [29]. This approach couples the network and the application processing in the same FPGA device. NARC [37] is a standalone network-attached FPGA board designed for high-performance computing and network applications. The board consists of a Xilinx FPGA and an ARM processor. The ARM processor is used as a network interface to connect the FPGA to the network via an Ethernet interface. IBM provides a new solution [38], which sets the FPGA free from the CPU to connect them directly to the data center network. This system implements 16 platforms of FPGAs interconnected together via an Ethernet switch. This deployment shows that the applications can scale the number of FPGAs independently from the number of severs. Hence, it improves the latency and throughput, respectively, by 40x and 5x.

FPGAs are deployed with CPU on data centers to accelerate the execution time and minimize the data center power consumption. In this case, cloud vendors are responsible for controlling the FPGA which hardware designers still do not have any access or control.

3. Challenges for FPGA in Local

A process of FPGA implementation needs several competencies and interactions. In this article, we will describe it by reference to two types of designers. An *application designer* is in charge of describing the algorithm, writing specifications, and validating the implementation, whereas a *hardware designer* is in charge of the FPGA design flow from creating the IP design to performing FPGA implementation on board as depicted in Figure 1. For complex architectures, several application designers and hardware designers collaborate to design and validate the architecture on FPGA. The application designer first describes the algorithm according to the application required. Specifications of the algorithm are then given to the hardware designer. The hardware designer selects IPs or designs Intellectual Property (IP) if not existing. Functions of the algorithm are implemented by IPs. IPs are designed with an HDL and simulated, optimized, and tested with FPGA design flow. The bitstream, the programming information of the FPGA, is generated, and the end of the design flow and the application designer can download this bitstream on the FPGA to validate and execute the algorithm. The time required between the specifications and the generated bitstream is long especially if IPs must be designed or the implementation constraints are hard.

There are some challenges from the FGPA and architecture design side (locally for the hardware designer) and for the validation of the application (locally for the application designer). These challenges are extracted from difficulties encountered in the FPGA architecture and design and in the validation of the application.

3.1. Challenges Related to FPGA Architecture and Design. The hardware designer creates the FPGA architecture with the FPGA design flow depicted in Figure 2 and specific FPGA tools integrated in the flow. The objective is to design, simulate, and test the generated architecture and also to modify the design to meet the resource and timing constraints. Some steps and tools in the design flow lead to two main challenges in the FPGA architecture and design:

The first challenge is linked to IP design (D1) and IP selection (D2). For IP design, the hardware designer describes the functions by means of IPs with hardware description languages. Such languages are completely different from languages used by application designers, and a strong hardware expertise is required. IP design is complex, requires IP competencies, and significantly increases the design cycle. Usually, several IP designers are required, and the continuous integration of designed IPs is done in the FPGA design flow, increasing more the design cycle. For IP core selection, the hardware designer selects the appropriate IPs corresponding to the functions of the application designer. The IP selection can be made amongst in-house IPs and third party IPs. For third party IPs, the hardware designer can buy different types of IPs (soft core, hard core, and firm core) from many specialized IP vendors. It is very common to get IPs from multiple vendors for a single design [84]. The IP selection depends on different criteria such as the IP functionality, portability, the FPGA resources, and the frequency required. "Off-the-shelf" IP is not necessarily optimized for the application, and IP licenses come with various restrictions such as reuse, disclosure, and rights modifications which can possibly impair the design flexibility. The first challenge from IP design and selection steps is called IPs store challenge (C1) where IPs in the architecture depend on many criteria like area, performance, and features, needed to be traded off against aspects like cost, license, risk, and time-to-market.

The second challenge mainly depends on tools and parameters. The FPGA design flow integrates several FPGA design tools. Required tools in the design flow are synthesis, simulation, and place-and-route tools. FPGA design tools selection (D3) is required according to the availability of tools, the prices, and the functionalities [40] of each of them. If several hardware designers are required, the FPGA design tools must be the same for all designers working on the same project. They should have also a strong expertise to use these tools. From these selected tools, another difficulty is the license software (D4). A same FPGA vendor offers several FPGA tools that can be integrated in the design flow. Buying an FPGA development kit enables the hardware designer to get specific licensed FPGA software with unlimited time and some specific functionalities. The hardware designer can also buy FPGA software without buying hardware platforms. He obtains the software with limited use in time. As an example, Intel provides to their hardware designers a set of FPGA development kit, the Stratix 10 GX, with one year license software for the associated tools, the Quartus Prime Pro design software. For a design with multiple designers, a floating point license with several seats is mandatory. For selected tools, a huge number of Tool parameters (D5) must

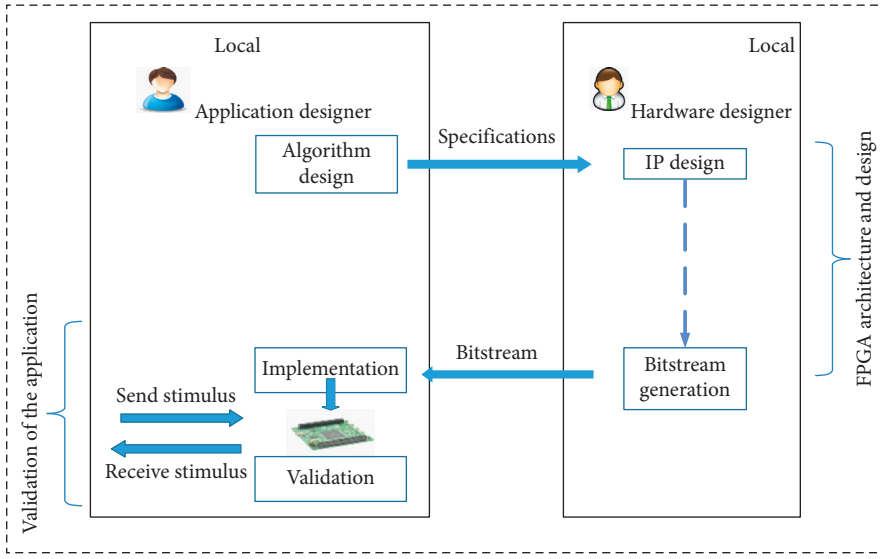

Figure 1: Interactions between application designers and hardware designers for FPGA design and implementation.

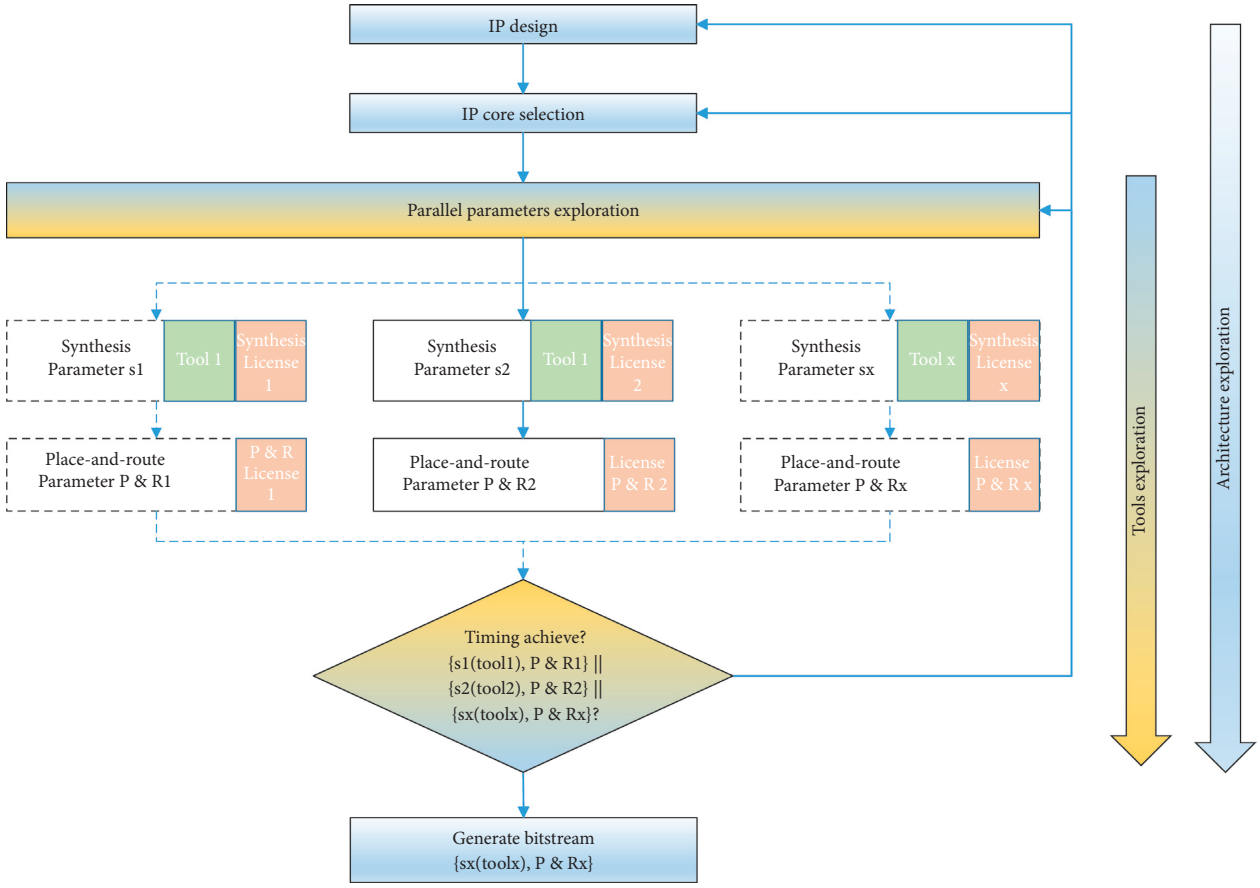

Figure 2: Example of FPGA design flow with parallel parameters exploration.

be set. Indeed, FPGA design tools offer a huge set of parameters to guide implementation results such as resources, speed, power, and many others constraints. The hardware designer can modify these parameters to obtain different performance results. For example, he can obtain different resource consumption balance at the synthesis level, a better timing result at the timing closure level, or encrypted bitstream to preserve confidentiality of implementation at the generated bitstream level. In [41], the author tests 1000 unique permutations of FPGA resources, called ALU4, along with the routed delay from the original netlist. The choice of parameters may significantly change according to the

requirements, and these parameters must be selected according to the targeted design. Another difficulty is timing closure (D6). This metric refers to the process by which an FPGA is modified to meet timing requirement. Modern FPGAs with the support of millions of LUTs and thousands of DSPs and internal block RAMs require several iterations and lots of CPU time to find the right combination of settings [42]. For instance, the learning-driven approach can be used to speed up convergence of timing closure [43]. The main challenge from all these encountered difficulties is the FPGA design tools and parameters (C2).

3.2. Challenges Related to the Validation of Application. The validation of the application consists in implementing the generated bistream on the FPGA, executing the application by sending and receiving stimulus. This step, called validation of the application, is performed by the application designer after receiving the bitstream provided by the hardware designer. The bitstream can be generated only if the application designer specifies the FPGA and the platform with the specification to the hardware designer. There are a large number of vendors selling FPGA devices and platforms. For the FPGA device selection (D7), the application designer needs to select the FPGA to use for the design by estimating the number and types of resources to be used and the operating frequencies of the design without having any hardware competencies. If the FPGA selected does not match with design requirements in terms of resource usage and operating frequency achieved, the hardware designer will not be able to implement the design on FPGA and the hardware designer will have to restart the synthesis and P&R steps. For example, in [39], authors evaluate the portability of IPs, IP SHA256, and Opencores AES 128 on three different FPGAs and prove that timing results depend on the FPGA device. The FPGA platform selection (D8) is required. FPGA platforms integrate one or multi-FPGA devices, external memories with different size, several communication protocols, and many other peripherals that can be used to execute the application. The application designer selects an FPGA platform according to FPGA devices and a set of external peripherals. The difficulty of choosing the right FPGA is that each new platform delivers more and larger memories and high number of inter-FPGA connections with different bandwidths. The performance of the design can significantly decrease for inappropriate components of the platform. The challenge is the FPGA and platform selection (C3) amongst all FPGA vendors and FPGA platform providers.

3.3. Summary of Difficulties and Challenges of Local Design and Validation. As presented previously, there are some difficulties leading to major challenges when designing FPGA in local and when validating the application in local. They are listed in Table 2.

4. Cloud FPGA Services

The cloud FPGA is an infrastructure of FPGA devices or software design tools available in the cloud. In order to extract the benefits of "putting FPGA in cloud," we propose to classify the cloud FPGA in three different levels of services:

(i) FPGA Software tools as a service: From the SaaS cloud model, FPGA tools are dematerialized since 2010 [44]. This model benefits from the massive computing of the cloud without worrying about tool incompatibilities and complicated setup steps. This model provides hardware designers an easy accessibility to the cloud without the complexity of infrastructure and software, but this model does not allow an access to a physical FPGA.

(ii) FPGA Platforms as a service: From the PaaS cloud model, FPGA platforms have been dematerialized before 2010 [45, 46]. There is no need to buy FPGA platforms. This model allows application designers to access to one or several FPGA platforms. The designer can develop and implement their cloud applications on old or newer FPGA platforms.

(iii) FPGA Resources as a service: From the IaaS cloud model, FPGA and its resources are virtualized since 2014 [47]. In this model, FPGA is divided into multiple independent virtual FPGA regions. These regions can be provisioned to multiple application designers in a multitenant environment with a virtual access to the physical FPGA.

This classification may change with time according to different criteria due to the advent of new applications or changes in requirements. According to the service proposed for the cloud of FPGA, previous challenges can be solved.

4.1. FPGA Software Tools as a Service. FPGA designs require very often several iterations before achieving timing closure. Synthesis and place-and-route tools include many parameters to control area and timing performances, and there is no one approach to achieve an optimized design. As designs are getting more and more complex, hardware designers require high compute capability to reduce design cycle, even with powerful workstations. With the adoption of management software by the industry such as Load Sharing Facilities (LSF) [48] or Sun Grid Engine (SGE) [49], FPGA design tools allow native control of computer bays. Quartus Design Space Explorer (DES), Xilinx Vivado, or ISE SmartExplorer use LSF to distribute multiple compilations. Cloud vendors support many versions of FPGA tools [50] (Quartus-II 13.0, Quartus Prime Standard/Pro Edition 15.1, SDSOC SDACCEL 2017.4, and Vivado 2017.2.1). We present major products that put the FPGA design flows and tools in the cloud without requiring any setup and maintenance for the cloud user.

To speed up the synthesis phase, Synopsys' Synplify Premier Synthesis tool uses the Common Distributed Processing Library (CDPL). Synplify Premier splits the design into submodules and uses CDPL to synthesize them separately. The result of each submodule merges into a single netlist, and the global report is available on Synplify Premier.

Figure 3 has an open view of all the components of the FPGA service. In local, the hardware designers design, select,

Table 2: Summary of challenges and difficulties of FPGA in Local.

Number of challenge	Type of challenge	Number of difficulty	Type of difficulty
C1	IP store	D1	IP design
		D2	IP selection
C2	FPGA design tool and parameters	D3	FPGA design tool selection
		D4	License software
		D5	Tool parameters
		D6	Timing closure
C3	FPGA and platform selection	D7	FPGA selection
		D8	Platform selection

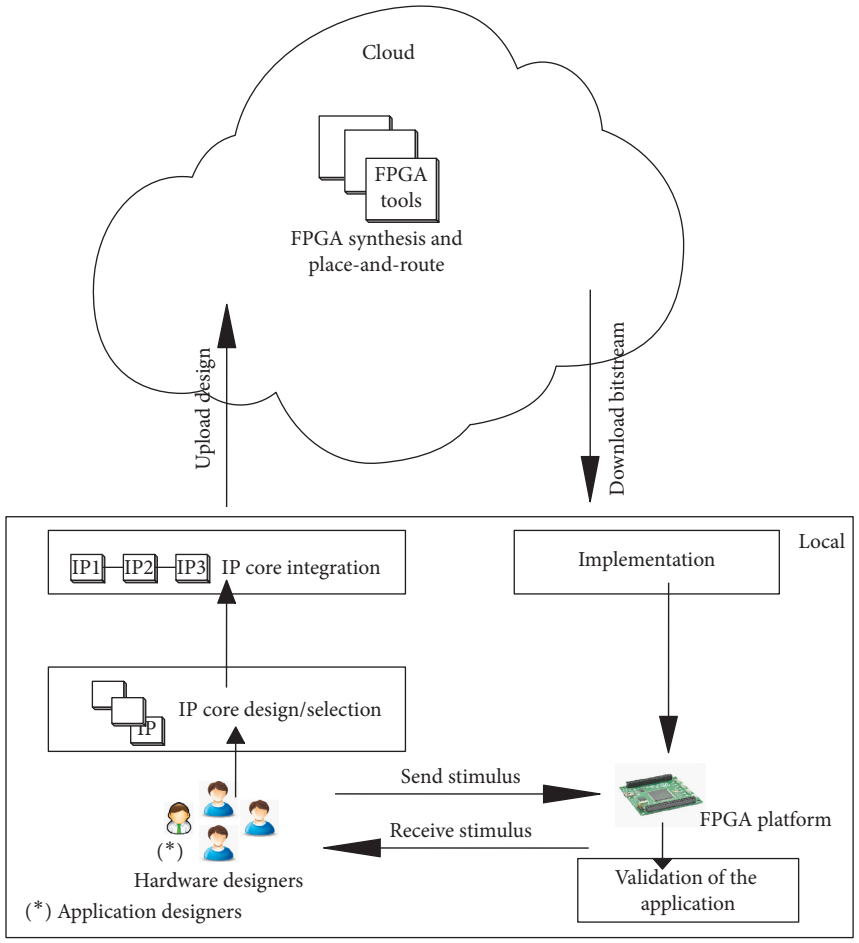

Figure 3: FPGA Tools as a Service model. The hardware designer accesses the cloud for all existing solution, except for the LabView tool, where the application designer can access the cloud.

and integrate IPs to design the FPGA architecture. They upload the FPGA design to the cloud. In the cloud, the existing FPGA tools synthesize and place and route the designs with several parameters of tools and for several FPGA devices and platforms if needed. FPGA design tools in the cloud enable the hardware designer to test a set of combination of parameters to explore the implementation results. The hardware designer delivers resource and timing results to the application designers for each generated bitstream. The application designer selects the most appropriate bitstream and downloads it in a local FPGA to validate the application. Plunify offers new services recently. The FPGA Expansion Pack [51] is a cloud plugin dedicated to the Vivado tool to implement different FPGA applications in multiple FPGA devices (D8) at the same time. Moreover, Plunify offers InTime [52], which is a machine learning tool dedicated to optimize RTL descriptions. With InTime service, the application designer can select a set of strategies such as the number of iterations and the number of servers. An iterative approach allows repeating the flow a configurable number of times. In the cloud, InTime distributes strategies in a huge compute farm of servers to optimize the timing closure (D6) [53] with a great number of parameters (D5). After optimization, the application designer downloads the result and validates the application. Kapre et al. [43] compare the results of InTime exploration with the

Altera Quartus DSE tool. They show that InTime is able to outperform higher timing score. The study demonstrates that the timing slack results are 7x better than Altera's DSE tool and for a set of opencores benchmarks. Furthermore, AI Lab [54] is a fully cloud-based virtual environment dedicated to develop machine learning and artificial intelligence applications dedicated to FPGA and GPU.

The NI LabView FPGA [55] of National Instruments allows application designers without any hardware expertise to implement and validate their applications in FPGA. The designer selects the NI platform that integrates an FPGA device and a Xilinx tool. The NI LabView automatically converts the schematic LabView to VHDL code. The application designer specifies a type and number of servers to implement his design in the cloud. The server distributes the VHDL files to available compilation machines. These machines perform the synthesis and place-and-route steps and generate a bitstream. The designer implements the bitstream in a local NI platform. This service is more dedicated to industrial companies that report the migration of FPGA tools to the cloud [56]. These companies provide different tools and devices for hardware designers with different contexts of use. Table 3 summarizes these tools provided as a service that meet two requirements. The first one is to exploit several FPGA tools (D3) distributed on multiple servers to simulate and evaluate several FPGA designs in parallel. The second one is to optimize designs and determines an efficient selection of synthesis and placement parameters. The FPGA software tools as a service are mainly dedicated to hardware designers. The hardware designer does not need to buy or maintain any FPGA tool license (D4), but he should test several versions of FPGA tools that run numerous iterations to get better optimization results. In most cases, the application designer is still depending on the hardware designer who develops and integrates IPs. Choosing and buying the suitable FPGA platform that meets the design requirements is still a big challenge.

4.2. FPGA Platforms as a Service. The issue associated with FPGA platforms is unavoidable as many academic institutions and companies have the problem of overcrowding. Several institutions are not properly funded; hence, they cannot provide the necessary number of FPGA platforms for every academic at the same time. In companies, some platforms are too big and too expensive to be deployed and provided for all designers. These issues (linked to D7 and D8) give birth to the idea of dematerializing FPGA platforms. Therefore, several institutions and companies have created their own remote FPGA labs. These labs offer a set of FPGA platforms with a remote visualization system [57], allowing application designers to quickly test and validate the new design.

Figure 4 explains how application designers can use the FPGA Platforms as a Service. In local, the hardware designers design their architecture with FPGA tools locally available. They generate the bitstream for the application designer. To access the cloud, the application designers upload the bitstream and validate the design by sending and receiving stimulus to the cloud. In most of cases, each FPGA is connected to webcams and other similar type of materials to visualize and manage FPGA platforms. Application designers can control the FPGA platforms and experiments with the webcam [58] in a real time. In local, application designers receive output stimulus and evaluate the resources and they can check the timing performances.

Soares et al. [59] present a simple approach for a remote laboratory using an Intel DE2 FPGA board including a Cyclone II device. The system targeted academics at introductory courses of digital design. ViciLogic [60, 61] is a remote academic platform which enables testing experiments over FPGA platforms. The ViciLogic has two versions. The most recent version (ViciLogic 2.0 prototype [62] is integrated in the Xilinx Vivado tool) automates online and local SoC digital logic hardware prototyping. It modifies the HDL model to provide signal observability and integrates SoC resources (ARM, AXI interconnect, peripherals, and viciLogic IP). Vicilogic 2.0 has generated two bitstreams: one in Xilinx Zynq SoCs and one in Intel CycloneV SoC. eDiViDe [63, 64] is a similar platform which hosts multiple FPGAs from different universities. Each FPGA is connected to a camera and/or microphone for registering the behaviour of the platform. Another recent academic remote lab [65] focuses its works in image processing such as the lane detection in road scenes applications. The application designers upload the bitstream generated via internet to the remote lab server. The server programs the FPGA with the bitstream and outputs the processed image. The remote lab allows application designers to compare resources used and power consumption on different types of FPGA platforms. Machidon et al. [66] use a single FPGA platform connected to the internet used for measuring functional parameters [67]. The hardware designer processes the design flow steps and generates the bitstream. Then, the application designer uploads the bitstream on the cloud and executes and validates the design. Several companies like Synopsys and Cadence provide large FPGA platforms dedicated for several complex application implementations. Zebu platform [68], HAPS [69], and proFPGA [70] are major existing hardware emulator platforms. As a result, the application designer uses one of these platforms remotely by using an Ethernet controller to benefit from highest performances and a low cost solution. Amazon provides an FPGA virtual machine image for its cloud FPGA instances, where users can easily develop and deploy FPGA acceleration applications [71]. Recently, Microsoft company presented AccelNet [72], which is an FPGA-based platform for host SDN processing supported by the software and the hardware infrastructure of the previous catapult project. Microsoft Azure also offers an FPGA-based platform to enable application designers to deploy machine learning applications. A key challenge is the multitenancy to efficiently share the FPGA while enforcing strict data and performance isolation between tenants. Alibaba [73] cloud officially launched three generations of large-scale FPGA instances, respectively, Ali F1, Ali F2, and Ali F3 based on Intel and Xilinx FPGAs to achieve strong isolation between IP acceleration and the deployment environment.

Table 3: Summary of FPGA software tools as a service.

Works	Description language[1]	FPGA tools[2]	Tool parameters[3]	Number of FPGA devices[4]	Implementation or optimization[5]	Sharing time[6]	Solved challenges[7]
FPGA expansion pack [51]	HDL	Xilinx Vivado	No parameter	Unknown limit	Implementation	Yes	C2* (i) License software (D4) (ii) Device selection (D7)
AI Lab [55]	HDL	Xilinx Vivado	No parameter	1	Implementation	Yes	C2*-C3* (i) FPGA tools selection (D3) (ii) Device selection (D7)
Intime [52, 53]	Netlist	Xilinx, Intel	Thousands of combinations	1	Optimization	Yes	C2* (i) Tool parameters (D5) (ii) Timing closure (D6)
Labview FPGA [55]	Labview code	Xilinx	Unknown limited to parameters of the FPGA tool selected	1	Implementation/ optimization	Yes	C2* (i) FPGA design tools selection (D3) (ii) License software (D4) (iii) Tools parameter (D5)

[1]Description language: referring to the input language required by the FPGA tools available in the service. [2]FPGA tools: providing the FPGA tool vendors in each service. [3]Tool parameters: offering the synthesis and place-and-route parameters that are offered by each service. [4]Number of FPGA devices: indicating the number of FPGA devices that they are available in FPGA tools. [5]Implementation or optimization: indicating the goal of each service. [6]Sharing time: ability of hardware designers to synthesis and place and route their applications with different FPGA tools at the same time. [7]Solved challenge; *indicates that the challenge is not fully solved and solved difficulties are listed.

Many remote labs exist and are mainly dedicated to academic institutions and industries in order to mutualize FPGA platforms and devices. Table 4 summarizes the projects of FPGA platforms in the cloud. Often remote FPGA lab provides a set of identical FPGA platforms with a limited number. The applications are executed in the cloud at the same time but for a limited time. While platforms are still valuable for small design to academic researches, industrial teams can use a powerful platform for debugging and implementing complex designs. These powerful platforms include a huge number of FPGA devices; therefore, multiapplication designers can implement their designs without worrying about the choice of FPGA device (D7) and platform (D8). In spite of these features, a designer must have a hardware expertise to develop designs and also to manage the FPGA tools locally.

4.3. FPGA Resources as a Service. The choice of an FPGA device has a huge impact on the resources and timing performances. It is very difficult to optimize the type of number of resources for one design. A solution to increase usage of resources is the virtualization of FPGAs. Such service provides FPGA resources instead of FPGA devices. Cloud vendors offer the use of diverse FPGA computing resources whose number varies according to the design. Figure 5 explains how an application designer can use the FPGA Resources as a Service. In local, the designers design their architecture with the FPGA design tools without specifying the FPGA device and the FPGA platform. The tools generate the bitstream for a set of FPGA resources. Each design generates a partial bitstream or a higher to target multiple FPGAs. To access the cloud, the application designers send the bitstream with specifying the FPGA resources that want to allocate. In the cloud, an hypervisor can configure several virtual bitstream with a partial FPGA region using the Openstack management system. The hypervisor informs the designer about the IP addresses of the virtual reconfigurable region. The Openstack must also track which virtual FPGAs or virtual FPGA regions have designer running in them and which FPGA belongs to which application designer. In local, the designer receives the resources and timing results of each region of the FPGA.

Existing works on FPGA virtualization can be classified in three levels [13].

4.3.1. Resource Level. A hardware resource on an FPGA can be either reconfigurable or nonreconfigurable. Hence, for this level, we consider architecture virtualization, called overlay architectures, and I/O virtualization. Existing works on FPGA virtualization are based on partial dynamic reconfiguration in order to maximize the usage of limited hardware resources by sharing several FPGAs. Many researchers use the OpenStack [74] to share FPGA resources to implement multiple designs on a single FPGA. The FPGA device is divided in multiple regions to support multiple designs each. Byma et al. [75] enable multiapplication

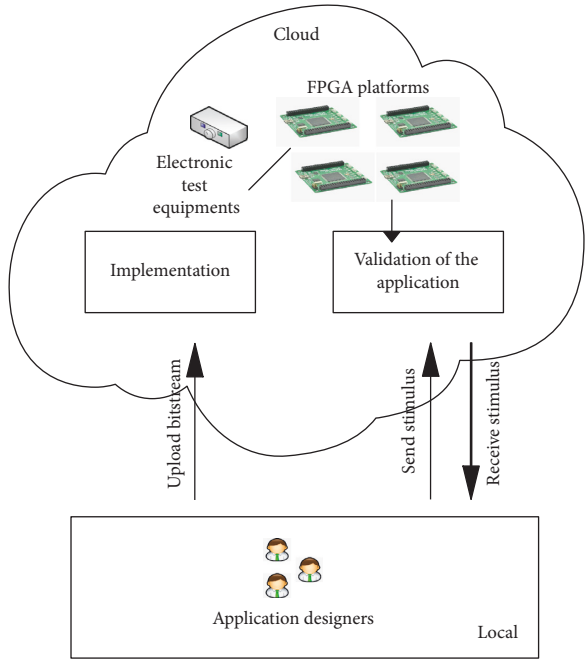

Figure 4: An FPGA Platform as a Service model.

Table 4: Summary of the FPGA platform as a service.

Works	Application[1]	Scalability of the cloud[2]	Heterogeneity[3]	Scalability of FPGA designs[4]	Multitenancy[5]	Sharing time[6]	Solved challenges[7]
Soares et al. [59]	Education	No[1]	No (Intel Cyclone II)	No	No	No	C3* FPGA device (D7)
Morgan et al. [60–62]	Education	Yes[2]	No (limited to one Xilinx Zynq SoCs and one Intel CycloneV SoC)	Yes	No	Yes	C3
Machidon et al. [66]	Education	Unknown limit	Yes	Yes	No	Yes	C3* FPGA
Machidon et al. [67]	Education	No[1]	No (just one Xilinx Spartan 3E)	Yes	No	Yes	platform selection (D7)
Zebu platform [68]	Industrial (emulation application)	Yes (limited to 64 maximum)	No (limited to Xilinx Virtex series)	Yes	No	Yes	C3* FPGA platform selection (D7)
HAPS platform [69]	Industrial (emulation application)	Yes (limited to 64 maximum)	No (limited to Xilinx Ultrascale series)	Yes	No	Yes	C3* FPGA platform selection (D8)
Microsoft [72]	Industrial	Yes	Yes (Intel Arria and Stratix Xilinx)	Yes	Yes		C3* FPGA platform selection (D8)
Amazon [71]	Industrial	Yes (one to eight)	No (limited to Xilinx Virtex UltraScale + VU9P or VU13P)	Yes	Yes	Yes	C3* FPGA platform selection (D8)
Alibaba [73]	Industrial	Yes	Yes (Intel Arria and Xilinx Ultrascale familties)	Yes (F1,2,3 instances FPGA)	Yes	Yes	C3* FPGA platform selection (D8)

[1]Application: referring to the target type of application, either academic or industry. [2]Scalability of the cloud: referring to adding or removing FPGAs in the cloud. [3]Heterogeneity: ability to use different types of FPGA in a same cloud. [4]Scalability of FPGA designs: ability of a hardware designer to target the designs to a multi-FPGA platform. [5]Multitenancy: ability of an FPGA to be used by multiple different hardware designers. [6]Sharing time: ability of application designers to execute their designs in different FPGAs in the same time. [7]Solved challenge; * indicates that the challenge is not fully solved and solved difficulties are listed.

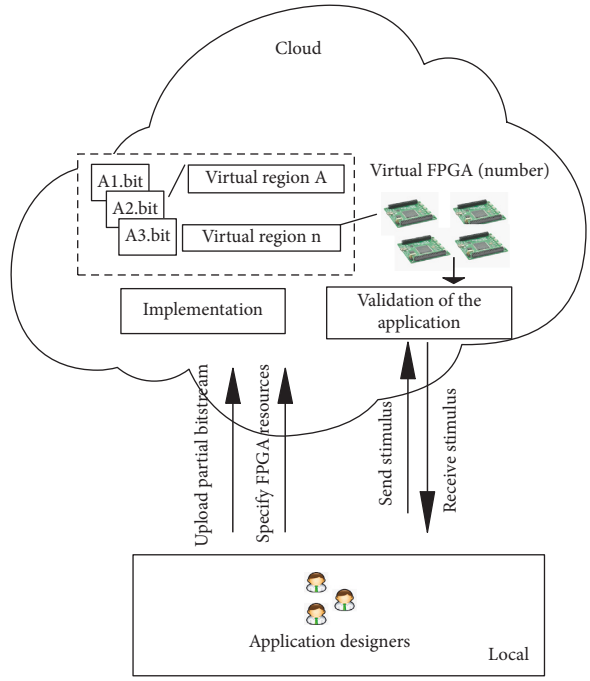

Figure 5: FPGA Resources as a Service model.

designers to virtualize and share memory on a single or multi-FPGA by accessing virtualized FPGA regions via an Ethernet connection. In this case, the control logic is implemented using a soft core, an embedded processor. The application designer programs the FPGA using an IP or MAC address of the FPGA or vFPGA. In another approach [76], authors aim at virtualizing nonreconfigurable FPGA resources. The Zeff platform enables a remote access for application designers to manage overlay architectures with the OpenStack. The architecture is synthesized using the Xilinx ISE and implemented in the Digilent Nexys3 FPGA board (Xilinx Spartan-6). The Zeff platform can deploy the same virtual bitstream on different FPGAs (Xilinx Artix 7 and CycloneV).

4.3.2. Node Level. One node is defined as a single FPGA. For this level, we consider infrastructure and resource management techniques.

Fahmy and Asiatici et al. [77, 78] propose an approach where applications designers can implement their design on one single FPGA partitioned into four reconfigurable regions. The designers share resources for simultaneous execution for one application. The control logic is implemented in a host CPU. Kidane et al. [79] aim at sharing resources among different virtualized IPs during runtime. The author proposes two cloud services to deploy a virtual IP core in a virtual FPGA. The Reconfigurable IP as a Service does not enable the hardware designer to access the virtual Reconfigurable Region. The Reconfigurable Regions as a Service enables the designer to access to virtualized reconfigurable regions. The hardware designer designs and tests his design with local FPGA design tools. Each design generates a partial bitstream. Once the application designer specifies the resources to use, partial bitstreams are uploaded in the OpenStack. After that, the designer can validate the design in a virtual reconfigurable region of FPGA. When partial bitstreams are allocated to FPGA regions, resources and timing results are provided.

4.3.3. Multinode Level. A multinode is defined as a cluster of more FPGAs. For this level, we consider techniques and architectures used to connect multiple FPGAs for accelerating a framework like InAccel [80]. Kirchgessner et al. [39] use a uniform hardware/software interface for multi-FPGA communications. The work is based on the portability of a set of open IP cores and tools across three platforms (GiDEL PROCStar III (4 FPGA Altera Stratix III), Pico Computing M501 (a single Xilinx Virtex-6), and Nallatech H101 (a single Xilinx Virtex-4)). This Virtual RC platform allows the application designer to synthesize the same design using ROCCC [81], Vivado HLS. This work is inappropriate for cloud data center because it is domain specific, and results require huge area and high delay overheads. In [82], Asghari et al. propose a scalable infrastructure based on HLS as a Service [83] for multiapplication designers. In the cloud infrastructure, there is an FPGA pool of several FPGA platforms, FPGA synthesis tools that are managed by a hypervisor. Authors use the Openstack to manage the high-level programs.

There are several methods of FPGA virtualization to share FPGA resources between multiapplication designers to reduce cost and enable the flexibility of the system. Table 5 summarizes several studies and surveys on FPGA Resources used as a Service with relevant metrics. At the hardware resource level, the reconfigurable resource is based on the FPGA architecture and the mapping of accelerators. Hence, the nonreconfigurable resource is based mostly on the I/O resources that exist in the CPU/software domain. At the

Table 5: Summary of FPGA resources as a service.

Works	FPGA virtualization level[1]	Partial region support[2]	Multitenancy[3]	Sharing space[4]	Elasticity of resources[5]	Solved challenges[6]
Byma et al. [75]	Resource level	Yes	Yes	No	Yes	C3* FPGA device selection (D7)
Fahmy et al. [78]	Node level	No	No	No	Yes	C3* FPGA device selection (D7)
Kirchgessner et al. [39]	Multinode	No	No	No	No	C3* FPGA platform selection (D7)
Najem et al. [76]	Resource level	Yes	No	No	No	C3* FPGA platform selection (D7)
Asiatici et al. [77]	Node level	Yes	Yes	No	Yes	C3* FPGA device selection (D7)
Kidane et al. [79]	Node level	No	Yes	No	Yes	C3* FPGA device selection (D7)
Dashtbani et al. [82, 83]	Multinode	No	Yes	No	No	C3* FPGA platform selection (D8)

[1]FPGA virtualization level: referring to the level of FPGA virtualization. [2]Partial region support: ability of the platform to support the partial reconfiguration. [3]Multitenancy: ability of multi-FPGA to be used by different applications. [4]Sharing space: ability of application designers to use many resources from different FPGA for one application. [5]Elasticity of resources: amount of resources that can be dynamically increased or contracted. [6]Solved challenge; *indicates that the challenge is not fully solved and solved difficulties are listed.

node level, the virtualization is based on the infrastructure that is required to manage the resources related to a single FPGA. The dynamic partial reconfiguration makes FPGA a multitenant device that can host multidesigners. Therefore, virtualized FPGA may have fewer resources than physical FPGA. Moreover, at the multinode level, acceleration works are performed across multiple FPGAs with a partial reconfiguration technique. Nevertheless, it is also still necessary for the hardware designer to set up and manage FPGA license tools to develop his design and generate the bitstream locally. Using FPGA Resources as a Service, the application designers implement and execute the algorithm without selecting the FPGA (D7) and the platform (D8).

5. General Discussion and Future Directions

We can now revisit the characteristics of cloud of FPGA and show the challenges that can be solved according to the proposed service. Some challenges can be fully tackled and some challenges are partially tackled by solving some difficulties related to challenges. The summary of the services is presented in Table 6. The first model is FPGA Tools as a Service which consists of industrial works that either implement FPGA designs in parallel, or try to optimize designs in cloud. The FPGA design flow can be a time consuming process. Therefore, this service avoids the hardware designer to predict in advance how many tools licenses to use and what parameters should be used. Hardware designers can test several FPGA tools without upfront software investment. This enables multiple application designers to choose the required FPGA device according to FPGA resources and timing performances. Furthermore, industrials leverage their analysis and optimization algorithms to reduce the implementation times and the development costs with providing an efficient selection of tools parameters.

The second model is FPGA Platforms as a Service which has dematerialized FPGA platforms to validate designs into an FPGA-based prototype. The FPGA Platform as a Service allows multiple application designers to test several platforms without buying any of them. The third model is FPGA Resources as a Service with the management and sharing of FPGA resources between multiple application designers. In this model, application designer can access virtual platforms to increase resources of FPGA device and to reduce the implementation time of the design. As demonstrated by the strengths attributed to these services, some drawbacks to design and use of FPGA may be solved by one service or other. However, even with the availability of on-demand tools, platforms, and FPGA resources in the cloud, the design and selection of IP have still not been envisioned. In all these services, the hardware designer is still in charge of developing IP or buying IPs from different vendors to integrate them in a system design. All drawbacks cannot be solved with the previously presented services. D1 and D2 drawbacks are not solved whatever the service used is.

The design of IPs remains difficult, and hence, in majority of cases related to FPGA experts have a strong expertise in IP design. Therefore, the challenge of IP store is not tackled whatever the service proposes, and an FPGA design always requires hardware competencies. Modifying IPs during the FPGA design flow will significantly make the

Table 6: Summary of the FPGA cloud services.

	FPGA tools as a service	FPGA platforms as a service	FPGA resources as a service
Application	Industrial	Industrial/Academic	Industrial/Academic
Cloud model support	SaaS	PaaS	IaaS
Cloud method	Dematerialization of FPGA tools	Dematerialization of FPGA	Virtualization of FPGA resources
Related works	[52–54, 57]	[59, 61–64, 66]	[69–73, 75, 76, 79]
Solved challenges[1]	C2*, C3*	C3	C3
Solved FPGA difficulties	(i) FPGA tools selection (D3) (ii) License software (D4) (iii) Tools parameters (D5) (iv) Timing closure (D6) (v) FPGA device selection (D7)	(i) FPGA platform selection (D8) (ii) FPGA device selection (D7)	(i) FPGA platform selection (D8) (ii) FPGA device selection (D7)
Open issues	(i) Lack of FPGA platforms, where hardware designers must buy FPGA platform (D8) (ii) Hardware designers must develop and integrate their own IPs locally (D1, D2)	(i) Application designers should define a fixed limited time when they use the platform (ii) Feasibility of using a set of FPGA on a sharing time with a low number of application designers (iii) Hardware designers must develop and integrate their own IPs locally (D1, D2) (iv) Hardware designers should setup FPGA tools locally to simulate and implement their designs (D3, D4, D5) (v) The hardware designer must achieve the timing closure (D6)	(i) Hardware designers must develop and integrate their own IPs locally (D1, D2) (ii) Hardware designers should setup FPGA tools locally to simulate and implement their designs (D3, D4, D5) (iii) The hardware designer must achieve the timing closure (D6)

[1]Solved challenges with * that the challenge is partially solved.

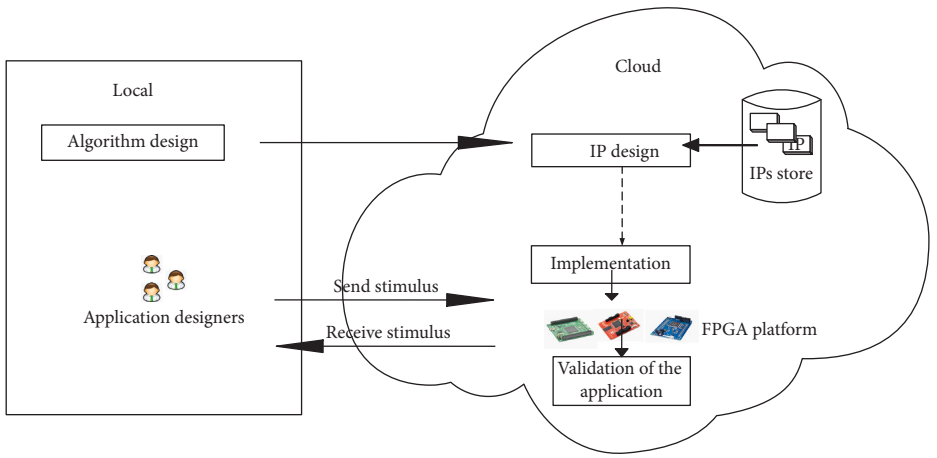

Figure 6: New service in the cloud of FPGA.

design process more complex and will increase the design cycle. The emerging interest in using FPGAs in the cloud represents the first widespread use of executing applications, and there remain numerous challenges to fully enable application designers to use IP cores. The trends towards more original systems in such context also present an opportunity well suited to execute applications. To truly resolve the difficulties of IP design (D1) and IP selection (D2) and tackle the **C1** challenge, we believe an infrastructure with IPs as a Service can tackle this challenge.

The proposed infrastructure is depicted in Figure 6. This model proposes a new service, which contains FPGA devices, platforms, and IP cores. FPGA design tools are not included in the infrastructure by default, but can be considered for a new IP provider in the cloud.

This cloud of FPGA is based on different levels:

(i) At the service level, how to better support the idea of providing a new hybrid service with an easy selection of IP cores.

(ii) In the programming model, finding the best programming model for interconnecting IP blocks. It is interesting to determine what type of communications should be used to connect IP cores.

(iii) At the designer level, explaining the removal of designer's interaction in both independent sides. The hardware designer is not in interaction with the application designer anymore and design cycle only depends on the application designer.

(iv) At the management level, it is an important open question to identify a control system that can automatically supervise the platform and can interact with the environment.

(v) Exploring how autonomously self-adaptive systems can be built that combine hardware designer capability with providing predesigned IP cores and the ability of application designers to execute their FPGA applications in cloud.

6. Conclusions

The paper presents an overview of the cloud FPGA concept that is based on the cloud computing paradigm. Using FPGA as a Service has emerged to become an active research which has attracted the attention of many researchers and industrials. In brief, we outline the starting of putting FPGA in data centers to support their infrastructure and to solve some cloud issues. Despite the major drawbacks of designing and using FPGA in local for one hardware designer, FPGAs present a compelling alternative for the cloud to be used as a service.

The purpose of this work is to provide a general use of FPGA to any designer (not domain specific) and support the pooling of tools, platforms, and resources. Three main classifications for cloud FPGA are discussed. The first classification extracts the context of SaaS through FPGA software tools. It addresses the migration of FPGA software tools to the cloud. Industrials allow designers to test several designs on several tools or to optimize automatically their designs. The second classification is developed from PaaS which consists of dematerializing FPGA platforms in the cloud. Using remotely large FPGA platforms allows application designers to control and monitor experiments during real time. The third classification extracts the context of IaaS. FPGA regions can be shared between application designers using the partial reconfiguration method to make a higher usage of the resources. There are nonsolved drawbacks that make cloud FPGA much more interesting, but to make it real designing of IP is still needed for application designers.

We will propose a new cloud FPGA platform which represents both an opportunity and a challenge for designers. The opportunity is to enable multiapplication designers to use IP cores as a Service without having to develop them. The challenge consists in finding a trade-off that guarantees the management and the execution of application with taking into account the mutualisation of IP between multiapplication designers.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of FPGAs,GPUs, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 47–56, ACM, Monterey, CA, USA, February 2012.

[2] B. Tippetts, D. J. Lee, K. Lillywhite, and J. Archibald, "Review of stereo vision algorithms and their suitability for resource-limited systems," *Journal of Real-Time Image Processing*, vol. 11, no. 1, pp. 5–25, 2016.

[3] G. Lacey, G. W. Taylor, and S. Areibi, "Deep learning on fpgas: past, present, and future," 2016, http://arxiv.org/abs/1602.04283.

[4] H. Al-Samarraie and N. Saeed, "A systematic review of cloud computing tools for collaborative learning: opportunities and challenges to the blended-learning environment," *Computers and Education*, vol. 124, pp. 77–91, 2018.

[5] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2011.

[6] M. D Ryan, "Cloud computing security: the scientific challenges, and a survey of solutions," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2263–2268, 2013.

[7] J. Weerasinghe, R. Polig, F. Abel, and Ch. Hagleitner, "Network-attached FPGAs for data center applications," in *Proceedings of the International Conference on Field-Programmable Technology (FPT)*, pp. 36–43, IEEE, Xi'an, China, December 2016.

[8] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Y. Zomaya, "Performance and energy efficiency metrics for communication systems of cloud computing data centers," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 738–750, 2017.

[9] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *Proceedings of the International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pp. 877–880, IEEE, Nagercoil, India, March 2012.

[10] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *Proceedings of the 26th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–10, Lausanne, Switzerland, August-September 2016.

[11] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: a review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, 2011.

[12] N. Mohammedali and M. O. Agyeman, "A study of reconfigurable accelerators for cloud computing," in *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control—ISCSIC'18*, p. 5, Stockholm, Sweden, September 2018.

[13] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on FPGA virtualization," in *Proceedings of the 28th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 131–1317, IEEE, Dublin, Ireland, August 2018.

[14] D. C. Le and C. H. Youn, "Criteria and approaches for virtualization on modern FPGAs," 2019, http://arxiv.org/abs/1904.03838.

[15] K. Vipin and S. A. Fahmy, "FPGA dynamic and partial reconfiguration: a survey of architectures, methods, and

applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 72, 2018.

[16] J. M. Mondol, "Cloud security solutions using FPGA," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 747–752, Victoria, Canada, August 2011.

[17] M. A. Will and R. K. L. Ko, "Secure FPGA as a service—towards secure data processing by physicalizing the cloud," in *Proceedings of the IEEE Trustcom/BigDataSE/ICESS*, pp. 449–455, Sydney, Australia, August 2017.

[18] M. F. Bari, R. Boutaba, R. Esteves et al., "Data center network virtualization: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[19] F. E. Mdarbi, N. Afifi, and I. Hilal, "Comparative study: dependability of big data in the cloud," in *Proceedings of the 2nd International Conference on Big Data, Cloud and Applications*, vol. 19, ACM, Tetouan, Morocco, March 2017.

[20] A. N. Toosi, R. N Calheiros, and R. Buyya, "Interconnected cloud computing environments: challenges, taxonomy, and survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 7, 2014.

[21] D. C. Marinescu, *Cloud Computing: Theory and Practice*, Morgan Kaufmann, Burlington, MA, USA, 2017.

[22] L. Ren, L. Zhang, L. Wang, F. Tao, and X. Chai, "Cloud manufacturing: key characteristics and Applications," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 6, pp. 501–515, 2017.

[23] M. U. Bokhari, Q. Makki, and Y. K. Tamandani, "A survey on cloud computing," in *Advances in Intelligent Systems and Computing*, vol. 654, pp. 149–164, Springer, Singapore, 2018.

[24] J. W. Rittinghouse and J. F Ransome, *Cloud Computing: Implementation, Management, and Security*, CRC Press, Boca Raton, FL, USA, 2016.

[25] A. Dutt, H. Jain, and S. Kumar, "Providing Software as a Service: a design decision(s) model," *Information Systems and E-Business Management*, vol. 16, no. 2, pp. 327–356, 2018.

[26] J. Moura and D. Hutchison, "Review and analysis of networking challenges in cloud computing," *Journal of Network and Computer Applications*, vol. 60, pp. 113–129, 2016.

[27] P. S. Suryateja, "Threats and vulnerabilities of cloud computing A review," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 3, pp. 297–302, 2018.

[28] S. S. Manvi and G. Krishna Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, 2014.

[29] A. M. Caulfield, E. S. Chung, A. Putnam et al., "A cloud-scale acceleration architecture," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*, p. 7, IEEE Press, Taipei, Taiwan, October 2016.

[30] B. S DJordjevic, S. P Jovanovic, and V. V Timčenko, "Cloud computing in Amazon and Microsoft Azure platforms: performance and service comparison," in *Proceedings of the 22nd Telecommunications Forum Telfor (TELFOR)*, pp. 931–934, IEEE, Belgrade, Serbia, November 2014.

[31] Amazon, "Amazon EC2 F1 instances," 2019, https://aws.amazon.com/fr/ec2/instance-types/f1/.

[32] J. Ch, R. Lian, Z. Li, A. Canis, and J. Anderson, "Accelerating Memcached on AWS cloud FPGAs," in *Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, p. 2, ACM, Toronto, Canada, June 2018.

[33] H. Schmit and R. Huang, "Dissecting Xeon+FPGA.: why the integration of CPUs and FPGAs makes a power difference for the datacentre," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 152-153, ACM, San Francisco, CA, USA, February 2016.

[34] A. Putnam, G. Jan, G. Michael et al., "A reconfigurable fabric for accelerating large-scale datacenter services," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 13–24, 2014.

[35] D. Chiou, "The microsoft catapult project," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, p. 124, IEEE, Seattle, WA, USA, October 2017.

[36] J. Weerasinghe, F. Abel, Ch. Hagleitner, and A. Herkersdorf, "Disaggregated fpgas: network performance comparison against bare-metal servers, virtual machines and linux containers," in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 9–17, IEEE, Luxembourg, UK, December 2016.

[37] C. Conger, I. Troxel, D. Espinoza, V. Aggarwal, and A. George, "NARC: Network-attached reconfigurable computing for high-performance, network-based applications," in *Proceedings of the Eighth Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD'05)*, Washington, DC, USA, September 2005.

[38] F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, and S. Paredes, "An FPGA platform for hyperscalers," in *Proceedings of the 25th Annual Symposium on high-performance interconnects (HOTI)*, pp. 29–32, IEEE, Santa Clara, CA, USA, August 2017.

[39] R. Kirchgessner, G. Stitt, A. George, and H. Lam, "VirtualRC: a virtual FPGA platform for applications and tools portability," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'12)*, pp. 205–208, ACM, Monterey, CA, USA, February 2012.

[40] N. Couture and K. B Kent, "Periodic licensing of FPGA based intellectual property," in *Proceedings of the IEEE International Conference on Field Programmable Technology*, pp. 357–360, IEEE, Bangkok, Thailand, December 2006.

[41] R. Y. Rubin and A. M. DeHon, "Timing-driven pathfinder pathology and remediation: quantifying and reducing delay noise in VPR-pathfinder," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'11)*, pp. 173–176, ACM, Monterey, CA, USA, February 2011.

[42] Beyond physical—solving high-end FPGA design challenges," 2009, https://www.synopsys.com/cgi-bin/proto/pdfdla/pdfr1.cgi?file=sbg-fpga-design-wp.pdf.

[43] N. kapre, B. Chandrashekaran, H. Ng, and K. Teo, "Driving timing convergence of FPGA designs through machine learning and cloud computing," in *Proceedings of the 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 119–126, IEEE, Vancouver, Canada, May 2015.

[44] Plunify, "Plunify," 2010, https://www.plunify.com/en/plunify-cloud/.

[45] R. Hashemian and J. Riddley, "FPGA e-lab, a technique to remote access a laboratory to design and test," in *Proceedings of the IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pp. 139-140, IEEE Computer Society, San Diego, CA, USA, June 2007.

[46] L. Gomes and S. Bogosyan, "Current trends in remote laboratories," *IEEE Transaction in Industrial Electronics*, vol. 56, no. 12, pp. 4744–4756, 2009.

[47] F. Chen, Y. Shan, Y. Zhang, Y. Wang et al., "Enabling FPGAs in the cloud," in *Proceedings of the 11th ACM Conference on Computing Frontiers*, p. 3, ACM, Cagliari, Italy, May 2014.

[48] IBM, "Introduction to LSF platform," 2012, https://www.ibm.com/support/knowledgecenter/en/SSETD49.1.3/lsffoundations/lsfintroductionto.html.
[49] UNIVA, "Gridengine," 2017, https://www.univa.com/products/.
[50] Plunify, "What are the supported FPGA tool versions?" https://support.plunify.com/en/knowledgebase/what-are-the-supported-fpga-tool-versions/.
[51] Plunifycloud, "Plunify cloud datasheet," https://support.plunify.com/en/doc/fpga-expansion-pack/.
[52] Plunifycloud, "Plunify cloud datasheet," https://docs.plunify.com/intime/.
[53] Q. Yanghua, N. Kapre, H. Ng, and K. Teo, "Improving classification accuracy of a machine learning approach for fpga timing closure," in *Proceedings of the IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 80–83, Washington, DC, USA, May 2016.
[54] Plunifycloud, "Plunify cloud datasheet," https://support.plunify.com/en/documentation/.
[55] NI, "Labview FPGA compile worker compile time benchmarks," http://www.ni.com/product-documentation/14040/en/.
[56] L. A. Dumitru, S. Eftimie, and C. Racuciu, "Using clouds for FPGA development-a commercial Perspective," *Journal of Information Systems and Operations Management*, vol. 11, no. 1, 2017.
[57] F. Morgan, S. Cawley, and D. Newell, "Remote FPGA lab for enhancing learning of digital systems," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 5, no. 3, p. 18, 2012.
[58] L. Bako, F. Morgan, S. Hajdu, S.-T. Brassai, R. Moni, and C. Enuachescu, "Development and embedded implementations of a hardware-efficient optical flow detection method," *Acta Universitatis Sapientiae Electrical and Mechanical Engineering*, vol. 6, pp. 5–19, 2014.
[59] J. Soares, J. Lobo, and FCT DEEC, "A remote FPGA laboratory for digital design students," in *Proceedings of the 7th Portuguese Meeting on Reconfigurable Systems, REC*, vol. 2011, pp. 95–98, Varanasi, India, January 2018.
[60] F. Morgan, S. Cawley, A. Coffey et al., "viciLogic: online learning and prototyping platform for digital logic and computer architecture," in *Proceedings of the eChallenges E-2014 Conference*, pp. 1–9, IEEE, Vilnius, Lithuania, November 2014.
[61] F. Morgan, S. Cawley, M. Kane, A. Coffey, and F. Callaly, "Remote FPGA lab applications, interactive timing diagrams and assessment," in *Proceedings of the 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pp. 221–226, Limerick, Ireland, June 2014.
[62] F. Morgan, D. O'Loughlin, J. Audiger et al., "Vicilogic 2.0: online learning and prototyping of digital systems using PYNQ-Z1/-Z2 SoC," in *Proceedings of the International Symposium on Rapid System Prototyping (RSP)*, pp. 76–82, Torino, Italy, October 2018.
[63] eDiViDe, "FPGA lab," 2013, http://www.edivide.eu/.
[64] J. Vandorpe, T. Bartkewitz, M. Drutarovsky et al., "eDiViDe: European digital virtual design lab-A remote learning platform for digital design," in *Proceedings of the SEFI Annual Conference, SEFI & KU Leuven*, Leuven, Austria, September 2013.
[65] M. Winzker, R. Kiessling, A. Schwandt, C. S. Paez, and S. A. Shanab, "Teaching across the ocean with video lectures and remote-lab," in *Proceedings of the IEEE World Engineering Education Conference (EDUNINE)*, pp. 1–4, IEEE, Buenos Aires, Argentina, March 2018.
[66] O. Machidon, F. Sandu, C. Zaharia, P. Cotfas, and D. Cotfas, "Remote SoC/FPGA platform configuration for cloud applications," in *Proceedings of the International Conference on Optimization of Electrical and Electronic Equipment (OPTIM)*, pp. 827–832, IEEE, Brasov, Romania, May 2014.
[67] O. M. Machidon, A. L. Machidon, P. A. Cotfas, and D. T. Cotfas, "Implementing a remote laboratory on a chip," in *Proceedings of the IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 155–158, IEEE, Constanta, Romania, October 2017.
[68] Synopsys, "ZeBu server 4: industry's fastest emulation system," https://www.synopsys.com/verification/emulation/zebu-server.html.
[69] Synopsys, "HAPS prototyping hardware," https://www.synopsys.com/verification/prototyping/physical-prototyping-hardware.html.
[70] Cadence, "Protium FPGA-based prototyping platform," https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/fpga-basedprototyping/protium-fpga-based-prototyping-platform.html.
[71] Bittware, "FPGAin the cloud: should you rent or buy FPGAs for development and deployment?" 2019, https://www.bittware.com/resources/fpgas-in-the-cloud/.
[72] D. Firestone, A. Putnam, S. Sambhrama et al., "Azure accelerated networking: SmartNICs in the public cloud," in *Proceedings of the 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pp. 51–66, Renton, WA, USA, April 2018.
[73] Alibaba, "FPGA-based compute-optimized instance families," https://www.alibabacloud.com/help/doc-detail/108504.htm.
[74] N. Tarafdar, T. Lin, D. Ly-Ma, D. Rozhko, A. Leon-Garcia, and P. Chow, "Building the infrastructure for deploying FPGAs in the cloud," in *Hardware Accelerators in Data Centers*, pp. 9–33, Springer, Cham, Switzerland, 2019.
[75] S. Byma, J. G. Steffan, H. Bannazadeh, A. L. Garcia, and P. Chow, "Fpgas in the cloud: booting virtualized hardware accelerators with openstack," in *Proceedings of the IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 109–116, IEEE, Boston, MA, USA, May 2014.
[76] M. Najem, T. Bollengier, J.-C. Le Lann, and L. Lagadec, "Extended overlay architectures for heterogeneous FPGA cluster management," *Journal of Systems Architecture*, vol. 78, pp. 1–14, 2017.
[77] M. Asiatici, N. George, K. Vipin, S. A. Fahmy, and P. Ienne, "Virtualized execution runtime for fpga accelerators in the cloud," *IEEE Access*, vol. 5, pp. 1900–1910, 2017.
[78] S. A. Fahmy, K. Vipin, and S. Shreejith, "Virtualized FPGA accelerators for efficient cloud Computing," in *Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 430–435, IEEE, Vancouver, BC, Canada, July 2015.
[79] H. L. Kidane, El-B. Bourennane, and G. Ochoa-Ruiz, "Noc based virtualized accelerators for cloud Computing," in *Proceedings of the IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-On-Chip (MCSOC)*, pp. 133–137, IEEE, Lyon, France, September 2016.
[80] InAccel, "Accelerated solutions for the cloud," 2018, https://inaccel.com/.
[81] J. Villarreal, A. Park, W. Najjar, and R. Halstead, "Designing modular hardware accelerators in C with ROCCC 2.0," in

Proceedings of the 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'10), pp. 127–134, IEEE Computer Society, Charlotte, NC, USA, May 2010.

[82] M. Asghari, A. Rajabzadeh, and M. Dashtbani, "HFIaaS: a proposed FPGA infrastructure as a service framework using high-level synthesis," in *Proceedings of the 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 72–77, IEEE, Mashhad, Iran, October 2016.

[83] M. Dashtbani, A. Rajabzadeh, and M. Asghari, "High level synthesis as a service," in *Proceedings of the 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 331–336, IEEE, Mashhad, Iran, October 2015.

[84] M. Gputa, "Using 3rd party IP in ASIC/SoC design," *Design and Reuse Journal*, 2019, https://www.design-reuse.com/articles/31313/using-3rd-party-ip-in-asic-soc-design.html.