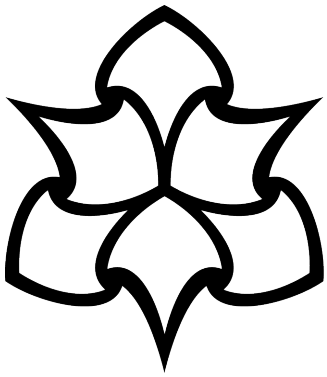

Manchester Metropolitan University

Ijaz, Q and Bourennane, EB and Bashir, AK and Asghar, H (2020) Revisiting the high-performance reconfigurable computing for future datacenters. Future Internet, 12 (4).

Downloaded from: http://e-space.mmu.ac.uk/627625/

Version: Published Version

Publisher: MDPI

DOI: https://doi.org/10.3390/FI12040064

Usage rights: Creative Commons: Attribution 4.0

Please cite the published version

https://e-space.mmu.ac.uk

future internet

MDPI

Review

Revisiting the High-Performance Reconfigurable Computing for Future Datacenters

Qaiser Ijaz [1,2,*], El-Bay Bourennane [1], Ali Kashif Bashir [3] and Hira Asghar [2]

1 ImViA Laboratory, University of Burgundy, 21000 Dijon, France; ebourenn@u-bourgogne.fr
2 Department of Computer System Engineering, Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan; hira.asghar@iub.edu.pk
3 Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M15 6BH, UK; dr.alikashif.b@ieee.org
* Correspondence: qaiser.ijaz@iub.edu.pk

Received: 2 February 2020; Accepted: 1 April 2020; Published: 6 April 2020

Abstract: Modern datacenters are reinforcing the computational power and energy efficiency by assimilating field programmable gate arrays (FPGAs). The sustainability of this large-scale integration depends on enabling multi-tenant FPGAs. This requisite amplifies the importance of communication architecture and virtualization method with the required features in order to meet the high-end objective. Consequently, in the last decade, academia and industry proposed several virtualization techniques and hardware architectures for addressing resource management, scheduling, adoptability, segregation, scalability, performance-overhead, availability, programmability, time-to-market, security, and mainly, multitenancy. This paper provides an extensive survey covering three important aspects—discussion on non-standard terms used in existing literature, network-on-chip evaluation choices as a mean to explore the communication architecture, and virtualization methods under latest classification. The purpose is to emphasize the importance of choosing appropriate communication architecture, virtualization technique and standard language to evolve the multi-tenant FPGAs in datacenters. None of the previous surveys encapsulated these aspects in one writing. Open problems are indicated for scientific community as well.

Keywords: FPGA virtualization; datacenters; network on chip; multi tenancy; multi FPGA; reconfigurable computing

1. Introduction

Today, datacenters are equipped with the heterogeneous computing resources that range from Central Processing Units (CPUs), Graphical Processing Units (GPUs), Networks on Chip (NoCs) to Field Programmable Gate Arrays (FPGAs), each suited for a certain type of operation, as concluded by Escobar et al. in [1]. They all purvey the scalability and parallelism; hence, unfold new fronts for the existing body of knowledge in algorithmic optimization, computer architecture, micro-architecture, and platform-based design methods [2]. FPGAs are considered as a competitive computational resource for two reasons, added performance and lower power consumption. The cost of electrical power in datacenters is far-reaching, as it contributes roughly half of lifetime cost, as concluded in [3]. This factor alone motivates the companies to deploy FPGAs in datacenters, hence urging the scientific community to exploit High-Performance Reconfigurable Computing (HRC).

Industrial and academic works both incorporated the FPGAs to accelerate large-scale datacenter services; Microsoft's Catapult is one such example [4]. Putnam et al. chose FPGA over GPU on the question of power demand. The flagship project accelerated Bing search engine by 95% as compared to a software-only solution, at the cost of 10% additional power.

The deployment of FPGAs in datacenters will neither be sustainable nor economical, without realizing the multi-tenancy feature of virtualization across multiple FPGAs. To achieve this ambitious goal, the scientific community needs to master two crafts, an interconnect solution preferably Network on Chip (NoC) as a communication architecture and an improved virtualization method with all the features of an operating system. Accumulating the state of the art in a survey can foster the development in this area and direct the researchers into more focused and challenging problems. Despite of the two excellent surveys, [5] in 2004 and [6] in 2018, former one categorized the FPGA virtualization as temporal partitioning, virtualized execution, and virtual machines, while, after fourteen years, the later one classified based on abstraction levels to accommodate the future changes, but the communication architecture or interconnect possibilities are not fully explored. To address this gap, an improved survey on FPGA virtualization is presented with the coverage of network-on-chip evaluation choices as a mean to explore the communication architecture, and commentary on nomenclature of existing body of knowledge. We revisited the network-on-chip evaluation platforms in order to highlight its importance as compared to bus-based architectures. We stretched our review from acceleration of standalone FPGA to FPGAs connected as a computational resource in heterogeneous environment. We attempted to create a synergy through combining three domains to assist the designers to choose right communication architecture for the right virtualization technique and, finally, share the work in the right language, only then, multi-tenant FPGAs in datacenters can be realized.

The remaining of the review paper is organized, as follows. Section 2 includes the commentary on nomenclature and recommendations for the scientific community. Section 3 talks about the available NoC evaluation tools to find out precise communication architecture in relatively less time. Section 4 puts the virtualization works into limelight with focus on architectures that are scalable and support multi-applications or multi-FPGAs. Section 5 indicates the trends and open problems as well presents a closing discussion about the area.

2. Revisiting the Nomenclature

The applications of FPGAs as computing resource are diverse that includes data analytics, financial computing and cloud computing. This broad range of applications in different areas requires efficient applications and resource management. This lays the foundation for the need of virtualizing the FPGA as a potential resource. Nomenclature is much varying due to the different backgrounds of the researchers contributing to this area. There are many such examples in literature where similar concepts or architecture is described using a different name or term. There is also an abundance of jargon terms and acronyms, which confuse the researchers rather enhancing their understanding. Table 1 identifies and lists non-standard terms in literature from the last decade.

Table 1. Non-Standard Nomenclature Present in Literature.

Year	Non-Standard Term(s) in Published Literature
2010	RAMPSoC in [7]
2011	Lightweight IP (LwIP) in [8]
2012	ASIF (Application Specific FPGA) in [9]
2013	sAES (FPGA based data protection system) in [10]
2014	PFC (FPGA cloud for privacy preserving computation in [11]
2015	CPU-Cache-FPGA in [12]
2016	HwAcc (Hardware accelerators), RIPaaS and RRaaS in [13]
2017	FPGA as a Service (FaaS) and Secure FaaS in [14]
2018	ACCLOUD (Accelerated CLOUD) in [15], FPGAVirt in [16]
2019	vFPGA-based CCMs (Custom Computing Machines) in [17]

This area is stagnated for a lack of a standard nomenclature. We recommend that the scientific community should use a unified nomenclature to present the viewpoint in order to improve the clarity and precision of communication for advancing the knowledge base. We also recommend that this area

must be referred as High-Performance Reconfigurable Computing (HRC) in literature. Moreover, it has been observed that the use of computer science language is more conveying as virtualization in FPGAs is comparable to an operating system in CPUs.

We urge the scientific community to come together to develop nomenclature, as it will improve the communication among researchers. It will ease the classification of works for entry-level researchers and help them to focus on complex research problems.

We acknowledge some quality examples such as the suitability of FPGAs has been discussed in depth in the context of high performance computing and heterogenous computing resources in [1], a new classification of FPGA virtualization has been presented in [5], and state of the art has been explored in the context of cloud computing, as defined by the National Institute of Standards and Technology in [18]. These authors have used the standard language of computer science and written in such a way that it added value to the understanding of readers.

3. Revisiting the Network on Chip Evaluation Tools

Data transfers in most of the high-performance architectures are limited by memory hierarchy and communication architecture, as summarized in [19,20]. Exploiting communication architecture suggests the use of NoC, an effective replacement for buses or dedicated links in a system with large number of processing cores [21,22]. NoC is composed of several tunable parameters like network architecture, algorithm, network topology and flow control. No System on Chip (SoC) is outright without NoC, today, due to promised high communication bandwidth with low latency as compared to the alternate communication architectures.

Researchers heavily rely on automated evaluation tools, where performance and power evaluation can be viewed early in design, given the complexity of NoC. Figure 1 describes a typical cycle of NoC evaluation, with FPGA being connected to a Central Processing Unit (CPU). Traffic scenarios are generated through traffic generator, sent to NoC that resides in FPGA, and the evaluation results are received through traffic receptors. Tools for FPGA based NoC prototyping are diverse architecture-wise. De Lima et al. in [23] identified an architectural model comprising of three layers: network, traffic, and management.

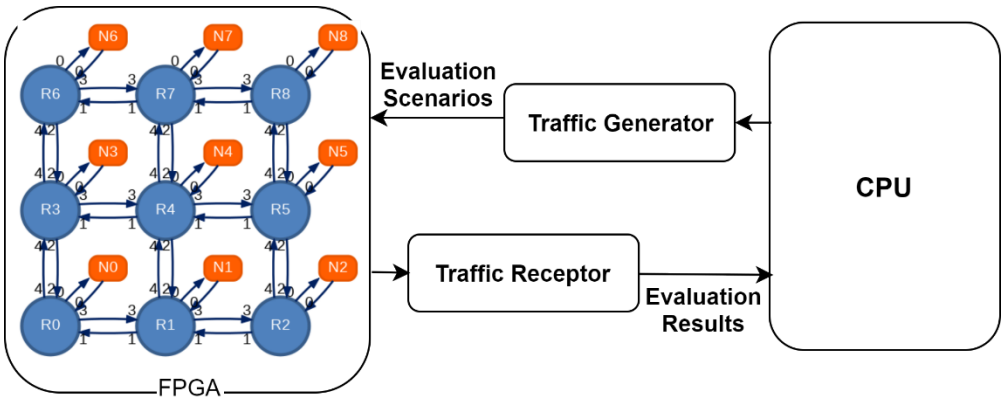

Figure 1. Generic Architecture of Networks on Chip (NoC) Evaluation on Field Programmable Gate Arrays (FPGA)(s).

There are four different types of network: Direct Mapping on Single or Multi FPGA(s), Fast Prototyping and Virtualization. The choice of the network affects the accuracy and resource utilization. Traffic on network can be generated in two different ways: synthetic and application-specific. Synthetic traffic is a kind of load testing to evaluate the overall performance, but it fails to forecast the performance under real traffic flow. Application-specific traffic, on the other hand, is based on the behavior of real traffic flow that is difficult to acquire but gives more accurate results. These patterns can be acquired either through trace, statistical method or executing application cores. As traces comprises

of millions of packets so the size becomes a limiting factor. Running application cores to generate traffic is also resource-expensive method.

Table 2 lists some FPGA based NoC evaluation tools, describing every architecture with network type, traffic type, number of routers, target board, and execution frequency, while hiding the complexity of NoC designs. The number of routers in NoC depends on the network type, architecture with relatively more routers, are based on second group type of network, fast prototyping and virtualization. We have used the direct mapping network type in our previous works due to relatively high execution frequency [24,25].

Table 2. NoC Evaluation Tools based on FPGA(s).

Year	Network Type	Traffic Type	No. of Routers	Target FPGA	Freq. (M.Hz.)	Work
2010	Multiple FPGA	Real: App. Cores	16	Virtex 5 × 5	-	[26]
2010	Direct Mapping	Real: Traces based	64	Virtex 2	45	[27]
2010	Fast Prototyping	Synthetic	49	Virtex 6	50	[28]
2011	Direct Mapping	Real: Traces based	25	Virtex 5	-	[29]
2011	Virtualization	Real: Traces based	256	Virtex 6	152	[30]
2011	Fast Prototyping	Synthetic	576	Virtex 6	300	[31]
2011	Virtualization	Real: App. Cores	64	Virtex 2	-	[32]
2011	Virtualization	Real: App. Cores	16	Virtex 5	3, 15	[33]
2011	Direct Mapping	Synthetic	36	Virtex 5	-	[34]
2012	Direct Mapping	Real: App. Cores	9	Virtex 5	-	[35]
2013	Multiple FPGA	Synthetic	18	Virtex 5 × 2	-	[36]
2014	Virtualization	Synthetic	1024	Virtex 7	42	[37]
2015	Direct Mapping	Synthetic	64	Virtex 6	50	[38]
2016	Direct Mapping	Synthetic	16	Virtex 6	250	[13]
2017	Direct Mapping	Synthetic	16	Virtex 6	250	[24]
2018	Direct Mapping	Synthetic	16	Virtex 6	250	[25]

These evaluation platforms assist the designers to reach the design-specific communication architecture, meeting most of the requirement specifications, for a certain application. These evaluation platforms take comparatively more time to synthesize the change, while on the other hand, a simulator can accommodate the same change in much lesser time. Designers offer dynamic reconfiguration, as a peroration to this limitation, but simulators are still the first choice of many entry-level researchers. However, the choice of NoC to realize the future datacenters with multi-tenant multi-FPGAs is yet to explore. The linking of several computational nodes becomes complicated and affects the performance of the overall system. Although NoC is not the only choice for communication within an FPGA as well as among multiple FPGAs but offer a competitive and promising solution. Other solutions include traditional bus, bus combined with a soft shell, different types of soft NoC and hard NoC. Many comparative studies evaluated these choices based on parameters like useable bandwidth, area consumption, latency, wire requirement and routing congestion. The way NoC is generated, also affects the performance so designers must be careful while choosing the NoC or an alternate for their design.

4. Revisiting the FPGA Virtualization

Resources are time multiplexed in a cloud services provider datacenter, referred as Infrastructure as a Service (IaaS). The sharing of resources is achieved through virtualization, an abstraction layer for hiding the physical resources from users. The process of virtualization raises issues like ease-of-use, privacy and performance but yet IaaS provide individual users and small organizations with an economic choice of renting over spending on infrastructure. Other than an academic example, such as SAVI testbed [39], industry offers plenty of solutions that are equally popular among designers. Amazon Web Services EC2 [40], IBM Zurich [41], and Intel are important competitors. Alveo on the Nimbix Cloud [42] is suitable for the designers working on Xilinx tools. Maxeler Technologies,

however, offers specific solutions, like an algorithmic contribution for memory mapping [43] and an area optimization technique [44].

Virtualization plays a relatable role to an operating system in a computer, but the term is being used in different meanings in this area, due to non-uniform nomenclature discussed earlier. Yet, the universal concept of an abstraction layer remains unchanged, a layer for the user to hide the underlying complexity of the computing machine, where the computing machine is not a traditional one, but FPGA. Many virtualization architectures have been proposed as per the requirements of the diverse applications. In 2004, a survey in this regard categorized the virtualization architectures into three broad categories, temporal partitioning, virtualized execution, and overlays [5]. Since then, no serious effort has been recorded on the classification of virtualization, until Vaishnav et al. [6] in 2018 classified the virtualization architectures based on abstraction levels. This much-needed classification contributed by Vaishnav et al. has been adopted as is, to discuss the works in this survey. We reiterated them with some of the representative work examples in Table 3. The works have been discussed under the same abstract classification.

Table 3. Classification of FPGA Virtualization adopted from [6].

Abstract Classification	Sub-Class	Work Examples References
Resource Level	Overlays	[45–74]
	Input Output (I/O) Virtualization	[4,75–80]
Node Level	Virtual Machine Monitors	[81–84]
	Shells	[4,41,75–78,85–97]
	Scheduling	[82,89,98–112]
Multi Node Level	Custom Clusters	[113–117]
	Frameworks	[77,103,118–123]
	Cloud Services	[4,40,124–130]

Although there are many features of virtualization like management, scheduling, adoptability, segregation, scalability, performance-overhead, availability, programmability, time-to-market, security, but the most important feature in the context of scope of this research is the multi-tenancy because it is essential for a sustainable and economically viable deployment in datacenters. FPGA has two types of fabric: reconfigurable and non-reconfigurable. The virtualization for the non-reconfigurable fabric is the same as of CPU, but there are several variations when it comes to the virtualization of the reconfigurable fabric.

4.1. Resource Level Virtualization

4.1.1. Overlays

Overlay architectures are diverse based on application and respective requirements. Overlays provide another higher abstraction layer on lower level fabric of FPGA, as depicted in Figure 2. The primary objective is to enhance the ease of programming for the software programmer. The reduced compilation time is an added advantage, given that the computer-aided design part to generate an accelerator is left out in the compilation process.

With respect to the ability of functional units, overlays are categorized in spatially-configured and time-multiplexed architectures. Li et al. compiled a comprehensive account of time-multiplexed overlays in a recent survey [45]. However, overlays are often discussed with respect to their implementation architectures in most of the literature that divides them into processor-based and coarse-grained reconfigurable architectures (CGRAs). A complete review of CGRAs can be found in Jain's doctorate thesis [46]. Processor-based comes in a variety of soft processor, either single-issue or multi-issue or multithreaded. They all add value to programmability, but the limited throughput is not suitable for the very high speed applications. Processor-based comes with a parallel processor as

well, either in the form or multithreaded or VLIW or soft vector processor or soft GPU. One form of the soft-core processors is [47], and similar solutions [48,49] are available from industry. Other forms include soft vector processors [50–54].

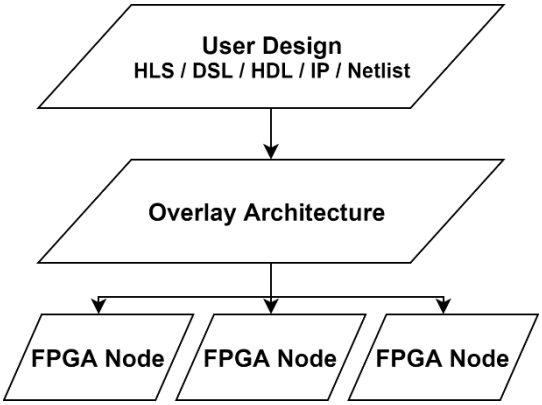

Figure 2. Generic Overlay Architecture.

CGRAs offer higher performance and scalability with lower power consumption, the very characteristics FPGAs are used for. CGRAs exist in the form of processing arrays or coarse to medium grained processing elements, where operations are performed at the processing element level. Examples of connected arrays of processing elements with programmable interconnects are [55–59]. Some CGRAs are kept dynamic by programming the processing elements and interconnect logic [60–63], while other architectures are kept static in spatial-configuration, as in [56,57]. Frequently appearing interconnect topologies in CGRAs are the nearest neighbor [55,56] and island model [57–59]. NoC are also found abundantly in CGRAs, some examples are [64–67]. NoC based architectures offer flexibility at the cost of higher implementation cost, but some works, like Hoplite soft NoC [67] and hard NoC [22], offer resource-efficient fast interconnects. Although an effort to reduce the cost by mapping the overlay look-up tables and multiplexers to the FPGA fabric has been achieved in [68]. Table 4 summarizes time-multiplexed CGRA overlays.

Table 4. Summary of CGRA based Overlays (*only industrial work).

Author(s) [Ref.] Year	Language	Board	Frequency	Granularity	Size
Ferreira et al. [69] 2011	VHDL	Virtex 6	100 MHz	8 / 32 / 64 bits	30
Kinsy et al. [70] 2011	Verilog	Virtex 5	155 MHz	32 bits	4×4
Brant [61] 2012	Verilog	Stratix III	150 MHz	32 bits	2×2
Paul et al. [63] 2012	-	Virtex 6	400 MHz	32 bits	40
Liu et al. [71] 2013	HLS Method	Zynq 7000	250 MHz	32 bits	2×2
Gray [72] 2016*	RISC V ISA	UltraScale	375 MHz	32 bits	$10 \times 5 \times 8$
Li et al. [73] 2016	HLS Method	Zynq 7000	286 MHz	32 bits	8
Kumar et al. [74] 2017	C	Stratix V	94 MHz	32 bits	60×2

Overlays are opted only to meet the requirement of rapid functionality change, where partial reconfiguration fails to cope with the speed of change, due to the sizable cost of implementation. Many solution providers have also commercialized this idea, like VectorBlox [50].

4.1.2. Input Output (I/O) Virtualization

I/O virtualization enables the access of different resources through the same interface or the sharing of resources among multiple applications. Figure 3 depicts the generic architecture where many virtual channels are represented with dashed lines, that do not equate to available physical channels. The middle layer in I/O virtualization plays multiple roles, like enforcing security mechanism,

monitoring resource utilization, ensuring Quality of Service (QoS) in datacenters, improving access time, and installing memory buffers.

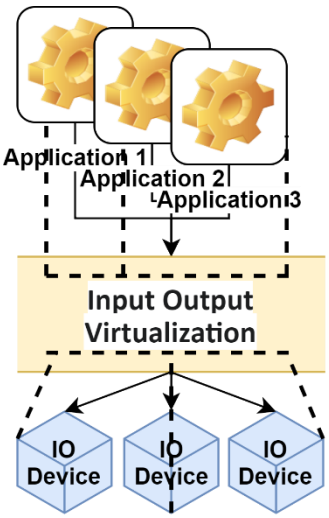

Figure 3. Concept of Virtual Channel in Input Output (I/O) Virtualization.

There are two possibilities for the design of control logic, either software or hardware. The software approach offers high flexibility and space efficiency [75–77]. On the other hand, the hardware module offers improved performance at the cost of consuming some reconfigurable resources [78,79].

Designers, as in [80], employed I/O virtualization to accelerate the storage up to 6x, beneficial for data intensive applications. Microsoft used the same to reduce the traffic of the network by directly handing over the requests to FPGA [4]. These are examples of the diverse use of I/O virtualization middle layer.

4.2. Node Level Virtualization

4.2.1. Virtual Machine Monitors (VMMs)

VMMs is the trouble-free method, as it takes many challenges away from FPGA. The scenario of treating FPGA as an attached peripheral to CPU provides the software programmers with multiple benefits, like familiar interface, libraries, and programming. Integrating accelerator with Virtual Machine Monitor (VMM) has almost zero performance overhead, as the experimental results showed in [81]. However, there are other approaches that further enhance the VMM capability to control many partial reconfigurable regions, like using micro-kernel [82], using micro-kernel to make a portable accelerator [83], and using OpenStack [84]. The idea of disassociating the static and dynamic fragments pays off at so many levels.

VMMs through resource allocation contribute to achieve many objectives for virtualization, such as multi-tenancy, management and scheduling, segregation, security, and availability. As FPGA is connected to CPU using standard frameworks, so multiple FPGAs can be added in the same arrangement.

4.2.2. Shells

Shells are referred as the static part of the system, which fundamentally provides the functionality of an operating system (OS); hence, various other names exist in literature, like FPGA operating system or hypervisor. It manages resources, I/O mechanism, required drivers and other essentials to configure or reconfigure the desired application. Figure 4 lays out the important infrastructures that have been proposed, developed, and tested. These architectures have been exhibiting a certain level of one or more virtualization characteristics and significant performance.

Multiple partial reconfiguration regions, symmetric and asymmetric, are being used for achieving multiple applications on a single FPGA. Symmetric or tiled regions are uniform in size, as in [85]. In this way, the resource allocation becomes flexible, as it can reside in one or more neighboring regions, which further minimizes the internal fragmentation, as in [78]. On the other hand, asymmetric regions support the modules of different sizes and save us from reconfiguring the whole FPGA [86] altogether.

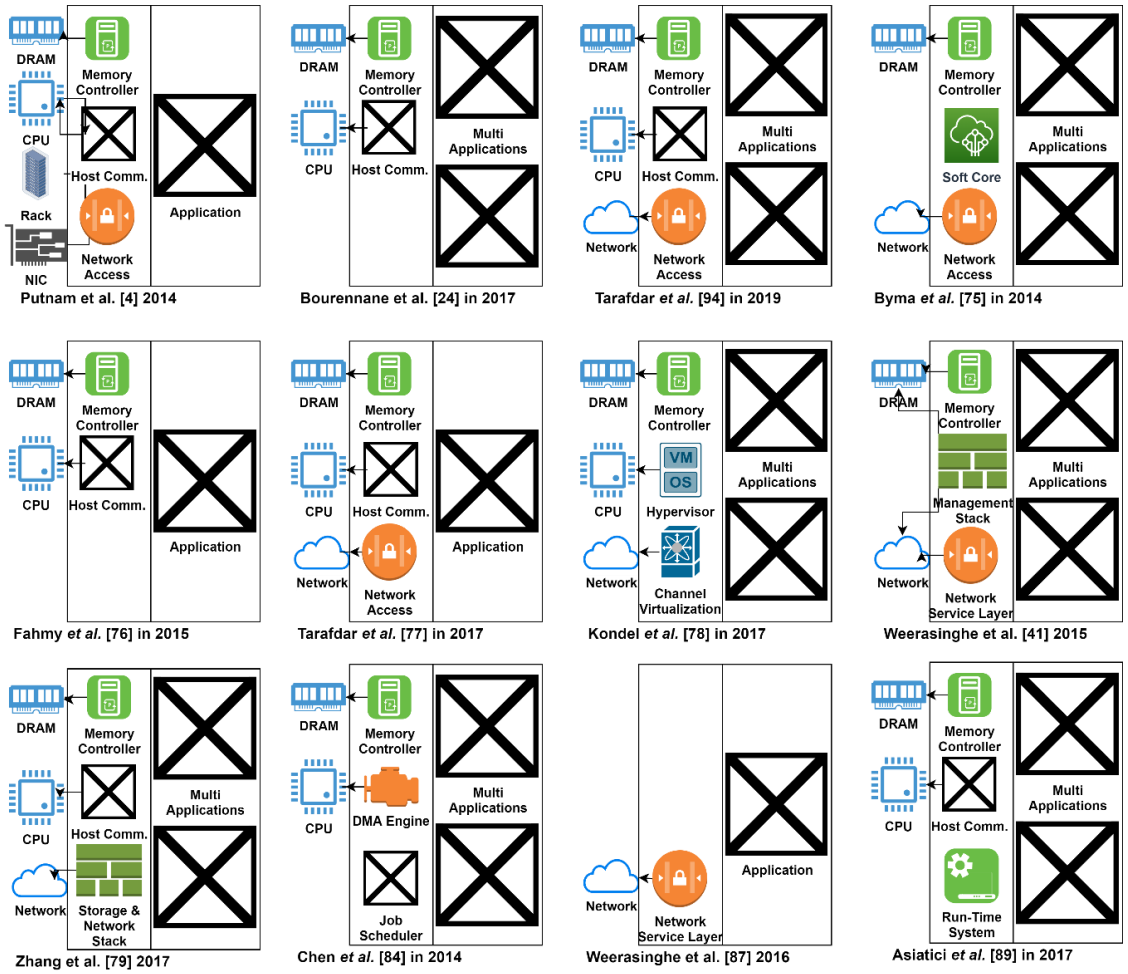

Figure 4. Possible Architectures for Utilization of FPGA(s) in Datacenter.

The connectivity is crucial for every execution model, it can either be host connectivity, or independent connectivity or the hybrid of the two. The architectures that are based on host connectivity only, CPU control the resources of the FPGA and reserve most of the reconfigurable resources and regions for the applications [76]. Multi-processors System on Chip (MPSoC) products from the FPGA vendors makes the implementation easier, however employing such products results in wastage of resources. Solutions, like [41,87], offer sharing among multiple CPUs, where [4,41,75,88] offer sharing among standalone FPGAs to avoid the underutilization of the FPGA resources. But the required support for the network layer consume a reasonable resource of FPGA. However, Asiatici et al. [89] developed a lightweight version featuring high-end application program interface (API), with a simpler execution model and shared memory. They proved their concept by measuring the marginal performance overhead. The hybrid approach offers more control intensive connectivity by exploiting offload to CPU, but additional hardware is required for I/O acceleration, as in [4]. Another type of shell called, container [90], is described as one without VMM, a process-level virtualization of application. This design has been accomplished by providing features like segregation, management and scheduling, and resolve for driver dependencies.

Considerable architectures have been tested in the last decade. The works that did impact the research in this area have been summarized in Table 5, along with hardware and virtualization characteristics.

Table 5. Research Highlights of the Decade.

Works	Hardware Characteristics			Virtualization Characteristics				
Authors, Reference, Year	Target FPGA Board(s)	FPGA Utilization	DDR Size (GB)	Partial Reconfiguration	Multitenancy	Scalability	Adoptability	Time to Market
Kirchgessner et al. [91] 2012	StratixIII, Virtex6, Nallatech H101	1% Area Overhead	-	No	No	Low	High	High
Byma et al. [75] 2014	Virtex5	74% BRAM	0.128	Yes	Yes	Med	Med	Med
Chen et al. [84] 2014	Kintex7	51% Logic	1.866	Yes	Yes	Med	Med	High
Putnam et al. [4] 2014	Virtex6, Stratix V	76% All	8	Yes	No	High	High	Low
Fahmy et al. [76] 2015	Virtex7	7% All	8	Yes	No	Low	Low	Med
Weerasinghe et al. [41] 2015	Zynq7100	33% Logic	-	No	Yes	High	Med	Med
Asghari et al. [92] 2016	Virtex7	-	-	No	Yes	Med	Med	Low
Bourennane et al. [13] 2016	Virtex6	11% LUTs	-	Yes	No	Med	Low	Low
Weerasinghe et al. [87] 2016	Virtex7	32% BRAM	8	Yes	No	High	Med	Med
Asiatici et al. [89] 2017	Virtex7	5% Area Overhead	8	Yes	Yes	Med	Low	Med
Kondel et al. [78] 2017	Virtex7	42% Logic	Virtualized	Yes	Yes	High	High	Med
Najem et al. [93] 2017	Artix7, CycloneV	30% FFs, 55% FFs	On Board	Yes	No	Low	Med	Med
Tarafdar et al. [77] 2017	Virtex7	20% BRAM	8	Yes	No	High	Med	High
Zhang et al. [79] 2017	StratixV	13% Logic	-	Yes	Yes	High	High	High
Bourennane et al. [25] 2018	Virtex6	3% LUTs	-	Yes	Yes	Med	Med	Med
Yazdanshenas et al. [22] 2018	Arria10	2% Logic	8	Yes	Yes	Med	High	Med
Tarafdar et al. [94] 2019	Kintex UltraScale	15–20% LUTs	-	No	Yes	High	Med	High
Vaishnav et al. [95] 2019	Zynq UltraScale+ Ultra96	12% LUTs 25% LUTs	2	Yes	Yes	High	High	High

Partial reconfiguration (PR) is used to reconfigure a part of FPGA dynamically, many architectures run more than one application using this function provided by the FPGA vendors. Multitenancy is defined as the capacity to serve multiple users using the same FPGA. Scalability is the qualitative measure of potential to scale up to multi FPGAs or multiple users with low overhead and congestion. Adoptability is featured as an acceptance of wide range of workload and applications, also referred as flexibility in previous works. Time-to-Market is a development time, directly the function of complexity of deployment on FPGA from design specifications. All these features of the shells are summarized in tabular form.

Industry also offers an API based solution, Intel's Open Programmable Acceleration Engine (OPAE) [96] is a collection of drivers, libraries, user and programmers' tools to enumerate, access, manipulate, and reconfigure programmable accelerators. Figure 5 provides detailed insight.

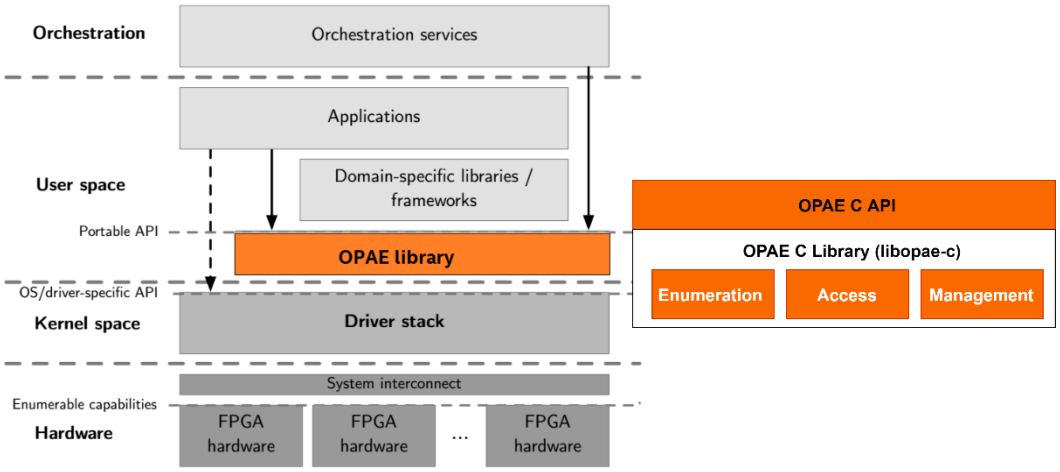

Figure 5. Open Programmable Acceleration Engine.

The designer must recognize that using shells can cause performance overheads due to layout limitations that are enforced by the partially reconfigurable slots. Furthermore, limiting the logic placement to a specific region on the chip can lead to longer wires that can result in slower modules [97]. Finding the optimal number of partial reconfigurable regions is a compelling open problem to explore, given the complexity of the shell and impact on the overall performance.

4.2.3. Scheduling

Scheduling is the key to multi-tenancy, but the conventional techniques (preemptive, non-preemptive, and cooperative) cannot be used for FPGA accelerators unchanged, as the state of the system that needs to be saved and restored is not trivial. The state data may be distributed across all different resources on FPGA fabric and one single operation to save or restore the state can add micro to milli seconds to the latency [98]. However, the requirement of mandatory dedicated hardware module can be avoided, as in [99], where such jobs are either blocked or sent back to CPU to perform.

The concept of scan-chain to provide the state data through an external interface, in order to make preemptive scheduling cost-effective and fast, has been implemented in [100] while using High-Level Synthesis (HLS) extension but for only a subset of registers. Non-preemptive scheduling has a low-cost implementation and a simpler design. Cooperative, on the other hand, only offer context switching at certain check points on the run time with the least overhead [82].

With the mature HLS methodology and availability of MPSoC platforms, hardware threads have been proposed as ReconOS [101] and Hthreads [102] with a pre-condition of tightly coupled CPU-FPGA to bring the scheduling closer to standard hardware description languages (HDLs).

Largely, scheduling techniques fall in non-preemptive category, which is fundamentally a time domain optimization. However, a dynamic approach has recently been introduced in [89], which takes advantage of the empty slots and keeps the utilization balanced on the run time. This dynamic scheduling technique enables the multi-tenancy like none other, as it gives the power of increasing or decreasing the resources usage, as per the workload requirement.

Some scheduling approaches are good for certain scenarios, like the one in [103,104], serve the multiple users at the same time without going through tedious partial reconfiguration, given that the accelerator needs of multiple users are the same. Another work, VineTalk [105], enables the FPGA sharing to a server or virtual machine in a datacenter, where the user has the liberty to choose through an API [106] among the GPU or FPGA accelerator, as per the need of the algorithm. In the heterogenous computing environment, OpenCL [107] is popular in practice and recommendation like SparkCL [108] solidified it further by bridging OpenCL and Java. OpenCPI [109] is an open source alternate of OpenCL. Important methodologies to mention are Intel HLS Compiler [110], Vivado High-Level Synthesis [111], and OpenSPL [112], as the programmability wall of FPGA remains a significant problem to this day.

With these platform and practices combined, the idea of future datacenters can be realized, as pictured in introduction. However, the process automation for the selection of appropriate accelerator in heterogenous computing environment is yet to be explored by the community.

4.3. Multi-Node Level Virtualization

The primary job is to distribute an acceleration job among multiple FPGAs, while abstracting the complex details from the user. The architecture of virtualization largely depends on how the multiple FPGAs are connected, there ways are depicted in Figure 6. This is not a standard, but the works so far have exhibited these formations. The direct model where FPGAs directly communicate with other FPGAs, where link represent the physical connection or virtualized I/O interface. The slave model where FPGAs are connected to the CPUs through PCIe or other links and CPUs are connected to the network, so if FPGA wants to send data to another FPGA, it goes through CPUs and network. The standalone model where FPGAs and CPUs are accessible though the network as standalone node. The designer can also combine them to form a hybrid model to meet the certain objectives.

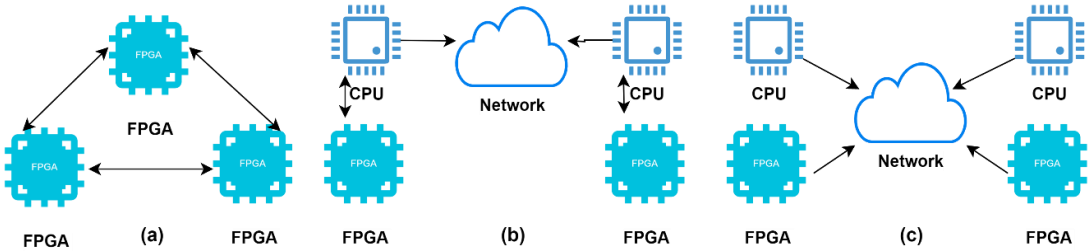

Figure 6. Multi-FPGA Architectures: (**a**) Direct Model represents FPGA to FPGA communication; (**b**) Slave Model represents FPGAs as special peripherals connected to Central Processing Units (CPUs) through PCIe; (**c**) Standalone Model represents accessible FPGAs, CPUs, or Graphical Processing Units (GPUs) through connected network.

Before discussing the sub classes, the salient features of some representative works of multiple FPGAs are to be discussed. Byma et al. [75] focused on the minimum virtualization overhead of medium scale datacenter providing commercial cloud services. They achieved significant performance when compared to regular virtual machines along with reduced iteration time for design. Kondel et al. [78] focused on maximizing the utilization of high-end FPGAs through paravirtualization and provided homogeneous virtualized FPGA regions for the clients. This flexible multi tenancy approach enables the individual resources to adopt the user requirements. Zhang et al. [79] developed an operating system to share single FPGA chip among different users at run-time with an improved resource manager. However, these mentioned works have not discussed the FPGA to FPGA or CPU connectivity in detail and the interfaces are not clearly described, except an indication of PCIe. Weerasinghe et al. [41] presented a different approach, FPGA as a standalone connected to datacenter network. The decoupled approach can utilize FPGA as an equal processing resource, especially in hyperscale datacenters. They chalked out a detailed system architecture with an outlook analysis on resource estimation and scaling perspectives.

A relatively recent trend is the emergence of tightly coupled CPU-FPGA platforms. Examples include Heterogeneous Architecture Research Platform (HARP) by the Intel and power chip combined with Coherent Accelerator Processor Interface (CAPI) by IBM. Academics responded to the call for proposals by Intel and several works have been published in last four years, some recent examples are [113,114].

4.3.1. Custom Clusters

Custom clusters are based on the concept of systolic array model in parallel computing architecture, where every node acts as a data processing unit and processed data move from

one node to another through first-in first-out (FIFO) buffer or network semantics. Some of these architectures [115–118] use Peer to Peer (P2P) connection MaxRing, fast series transceivers with FIFO buffers, and Peripheral Component Interconnect Express (PCIe) links, for transmitting data across multiple nodes. Tailored designs allow the direct communication among the nodes through explicit network connections. A cluster of 512 FPGAs [119] exploits the systolic array model to perform computations on multiple FPGAs.

4.3.2. Frameworks

Frameworks exploit the conventional server-client architecture, where only the computational part is assigned to one or more FPGAs, but the CPU server manages the rest, including configuration, application related data, and scheduling. The central piece in this architecture is the data management model, and models for CPU are equally extendible to FPGAs. For example, the idea of the MapReduce framework has been extended on FPGAs where mapping and reduction operations are performed by FPGA accelerators [103,118,119] in similar way as CPU client-server architecture. These frameworks have an added advantage of bridging the gap between the heterogeneity of datacenters, [120] is one such cluster comprised of FPGAs and GPUs while using MapReduce. Furthermore, Chen and colleagues in [104] extended java virtual machine (JVM) framework using Apache Spark to accommodate the FPGAs, this however comes with a communication overhead and requires precision.

Tarafdar [77] and his colleagues utilized OpenCL via Xilinx SDAccel framework using an abstract layer to assign the data to multiple FPGAs and maintaining a transparent directory to virtualize the FPGAs at the lower abstraction level. The approach of the FPGA groups [121] suggests that multiple FPGAs can be shared by one group but configured with a matching accelerator. However, this comes with a limitation of occupying a complete FPGA that results in under-utilization but it can be addressed with an automation of the scaling algorithm. A similar concept has been proposed in [122] while using Hadoop YARN with a value-added advantage of ease of programming.

In the heterogeneous computing environment, the performance is also a function of execution strategy. For the exploration of alternative execution strategies on disaggregated environments, the evaluation platform presented in [123] is useful.

4.3.3. Cloud Services

Cloud services architecture guarantees QoS and promises computational correctness while abstracting the underlying architecture. Therefore, as the user has no concern about the choice of computational node, the job can be computed on an employed FPGA. Amazon offering FPGA as a resource in [40] does not fall into this category but the landmark work of Microsoft [4] on search ranking that achieved a substantial speed-up, with relatively higher power consumption. This is also a good example of hybrid architecture as Catapult can allow for the acceleration jobs to both, host CPU core, and standalone FPGA. Baidu [124] achieved the same performance for deep neural networks. The use of FPGAs as co-processors in compute-intensive problems has been implemented [125], exploiting the multiple data streams.

The architectures with network support widen the choice of connectivity, which allows the CPU provisioning either as a soft-core or embedded on-chip. OpenStack is the most common method for directly allowing the user to program the FPGA [75,77,87] through physical or virtual address. It provides the flexibility to the expert user for exploiting either socket or remote routine approach to establish connectivity to an FPGA. Bashir et al. addressed the issue of poor utility and high computation complexity on high-dimensional data in [126] and proposed many network architectures for datacenters in [127,130,131].

4.4. Execution Model based Distribution

The execution model is used as a decision parameter while doing system partitioning, process to place certain modules in shell. Inspired from the Flynn Taxonomy in [132], the execution models can

be categorized as four, as described in Table 6. All of the works in the last decade have been distributed in any of the four boxes, as per relevance, for quick navigation.

Table 6. Distribution of Works based on Execution Model.

	Single Application	Multiple Applications
Single FPGA	[4,76,77,86,87,106,107,109,115,117,118,125]	[23,25,75,78,79,84–86,89,107]
Multiple FPGAs	[4,77,87,106,115,117,125]	[41,75,78,79]

5. Open Problems and Discussion

There is plenty to do in this area, but we would like to mention a few open problems. The foremost goal is to enable multi-tenant multi-FPGAs for medium to large-scale datacenters, only then we can unleash the real potential of FPGAs as a heterogeneous computing resource. This can be achieved either by developing FPGA operating system or improving existing virtualization methods. An intense investigation is required on how to compute over multiple FPGAs in a scalable manner. A design is required that can exploit multiple FPGAs via streaming between Catapult style or batching MapReduce style, other than OpenStack.

A serious effort is required to make the shell and development stack modular. Currently, everything must compile against a shell and any change in shell requires recompiling accelerators. Likewise, a change in the Linux kernel means the recompilation of all user software, so one can imagine how bad is the FPGA ecosystem yet today. However, FPGAs provide a lot of customization, without which it would be meaningless to use FPGAs in the first place. Overlays solve this issue to an extent for a small class of application, but the solution is not scalable for general computation with FPGAs. Therefore, we need a set of APIs and standards in software stack to manage this heterogeneity in a sensible manner. Dynamic resource allocation somehow addresses this issue, but largely it remains an ignored area by the community.

With multi-tenant support, efficient management and scheduling is required for the resources, an advanced resource manager that can fit the same workload on fewer FPGA resources should be the key point of future development.

Security is another aspect that needs intense attention of the community with a lot of potential for development. The complex case of FPGAs in datacenter is vulnerable to all sorts of attacks, as the reported attacks include malicious bitstream and side channel that severely damage the availability. It also assists in segregation of many accelerators on same FPGA or network.

6. Conclusions

The integration of FPGAs in datacenters might have different motivations from acceleration to energy efficiency, but the ultimate objective of better performance remained unshaken. FPGAs are being utilized in a variety of ways today, tightly coupled with heterogenous computing resources and a standalone network of homogenous resources. Open source software stacks, propriety tool chain, and programming languages with advanced methodologies are hitting hard on the programmability wall of the FPGA. Therefore, it was important to visualize this area as high-performance reconfigurable computing.

In this paper, we rendered a survey on high-performance reconfigurable computing. We pointed out the use of non-standard nomenclature in published research as an obstacle to the growth of the body of knowledge. We further identified, the contributors of different background, approaching for a wide range of applications, to be the reason of this phenomenon. We indicated some examples of using standard language and nomenclature. We revisited the network-on-chip evaluation platforms to highlight its importance as compared to the bus-based architectures. The limitations of virtualization shells like frequency drop, high wire demand, increased design latency, and routing congestion leading to routing failure, can be addressed using a suitable network-on-chip. We highlighted the need of network-on-chip evaluation platforms to quickly analyze the performance to reach a required

communication architecture. We updated the scientific community on classical and recent virtualization techniques, from the last decade. We stretched our review from acceleration of standalone FPGA to FPGAs that are connected as a computational resource in heterogeneous environment. The purpose of this research was to create a synergy through combining three domains to assist the designers to choose right communication architecture for the right virtualization technique and to emphasize the importance of using the standard language, so that multi-tenant FPGAs in the datacenters can be evolved.

We have chalked out open problems in this area. Our future research will be focused on finding optimal communication architecture, for multi FPGAs. Other than the interconnection between different processing elements within one FPGA, the communication among multiple FPGAs poses a bigger challenge in our future work, and an opportunity for the community as well.

Author Contributions: Conceptualization, Q.I. and E.-B.B.; methodology, A.K.B.; investigation, Q.I.; data curation, H.A.; writing—original draft preparation, Q.I.; writing—review and editing, A.K.B. and H.A.; supervision, E.-B.B.; funding acquisition, Q.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank Higher Education Commission, Government of Pakistan for PhD grant of the corresponding author of this paper. The authors would also like to thank Usman Ahmad, Dalhousie University of Canada for his advice on research writing in general.

Conflicts of Interest: The authors declare no conflict of interest regarding the publication of this paper.

References

1. Escobar, F.A.; Chang, X.; Valderrama, C. Suitability analysis of FPGAs for heterogeneous platforms in HPC. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 600–612. [CrossRef]
2. De Bernardinisa, L.P.; Pinelloa, C.; Sgroia, A.L. Platform-based design for embedded systems. In *Embedded Systems Handbook*, 1st ed.; CRC Press: San Francisco, CA, USA, 2005.
3. Inta, R.; Bowman, D.J.; Scott, S.M. The chimera: An off-theshelf CPU/GPGPU/FPGA hybrid computing platform. *Int. J. Reconfigurable Comput.* **2012**, *2012*, 241439. [CrossRef] [PubMed]
4. Putnam, A.; Caulfield, A.M.; Chung, E.S.; Chiou, D.; Constantinides, K.; Demme, J.; Esmaeilzadeh, H.; Fowers, J.; Gopal, G.P.; Gray, J.; et al. A reconfigurable fabric for accelerating large-scale datacenter services. In Proceedings of the 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), Minneapolis, MN, USA, 14–18 June 2014.
5. Plessl, C.; Platzner, M. Virtualization of hardware-introduction and survey. In *ERSA*; CSREA Press: Las Vegas, NV, USA, 2004.
6. Vaishnav, A.; Pham, K.D.; Koch, D. A survey on FPGA virtualization. In Proceedings of the 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, Ireland, 27–31 August 2018.
7. Göhringer, D.; Hübner, M.; Hugot-Derville, L.; Becker, J. Message passing interface support for the runtime adaptive multi-processor system-on-Chip RAMPSoC. In Proceedings of the 2010 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, Samos, Greece, 19–22 July 2010; pp. 357–364.
8. Shen, X.; Wang, X.; Zhu, Y.; Huang, T.; Kong, X. Implementing dynamic web page interactions with a Java processor core on FPGA. In Proceedings of the Engineering and Industries (ICEI), 2nd International Conference on IEEE, Jeju, Korea, 29 November–1 December 2011.
9. Farooq, U.; Parveza, H.; Mehrez, H.; Marrakchi, Z. A new heterogeneous tree-based application specific FPGA and its comparison with mesh-based application specific FPGA. *Microprocess. Microsyst.* **2012**, *36*, 588–605. [CrossRef]
10. Chen, Y.; Wang, Y.; Ha, Y.; Felipe, M.R.; Ren, S.; Aung, K.M.M. sAES: A high throughput and low latency secure cloud storage with pipelined DMA based PCIe interface. In Proceedings of the 2013 International Conference on Field-Programmable Technology (FPT), Kyoto, Japan, 9–11 December 2013; pp. 374–377.

11. Xu, L.; Shi, W.; Suh, T. PFC: Privacy preserving FPGA cloud—A case study of mapreduce. In Proceedings of the IEEE International Conference on Cloud Computing, Anchorage, AK, USA, 27 June–2 July 2014; pp. 280–287.
12. Yang, H.; Yan, X. Memory coherency based CPU-Cache-FPGA acceleration architecture for cloud computing. In Proceedings of the Information Science and Control Engineering (ICISCE), 2nd International Conference, Shanghai, China, 24–26 April 2015; pp. 304–307.
13. Kidane, H.L.; Bourennane, E.B.; Ochoa-Ruiz, G. Noc based virtualized accelerators for cloud computing. In Proceedings of the IEEE 2016, 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC), Lyon, France, 21–23 September 2016; pp. 133–137.
14. Will, M.A.; Ko, R.K. Secure FPGA as a service—Towards secure data processing by physicalizing the cloud. In Proceedings of the IEEE Trustcom/BigDataSE/ICESS 2017, Sydney, NSW, Australia, 1–4 August 2017; pp. 449–455.
15. Yazar, A.; Erol, A.; Schmidt, E.G. ACCLOUD (Accelerated CLOUD): A novel FPGA-Accelerated cloud architecture. In Proceedings of the 26th Signal Processing and Communications Applications (SIU), Izmir, Turkey, 2–5 May 2018; IEEE: Piscataway, NJ, USA, 2018.
16. Mbongue, J.; Hategekimana, F.; Tchuinkou Kwadjo, D.; Andrews, D.; Bobda, C. FPGAVirt: A Novel Virtualization Framework for FPGAs in the Cloud. In Proceedings of the IEEE 11th International Conference on Cloud Computing, San Francisco, CA, USA, 2–7 July 2018.
17. Al-Aghbari, A.A.; Elrabaa, M.E.S. Cloud-based FPGA custom computing machines for streaming applications. *IEEE Access* **2019**, *7*, 38009–38019. [CrossRef]
18. Skhiri, R.; Fresse, V.; Jamont, J.P.; Suffran, B.; Malek, J. From FPGA to support cloud to cloud of FPGA: State of the art. *Int. J. Reconfigurable Comput.* **2019**, *2019*, 8085461. [CrossRef]
19. Bittner, R.; Ruf, E.; Forin, A. Direct GPU/FPGA communication via PCI express. *Cluster Comput.* **2013**, *17*, 339–348. [CrossRef]
20. Mueller, R.; Teubner, J.; Alonso, G. Streams on wires: A query compiler for FPGAS. *Proc. VLDB Endow.* **2009**, *2*, 229–240. [CrossRef]
21. Dally, W.J.; Towles, B. Route packets, not wires: On-chip interconnection networks. In Proceedings of the Design Automation Conference, Las Vegas, NV, USA, 18–22 June 2001; pp. 684–689.
22. Yazdanshenas, S.; Betz, V. Interconnect solutions for virtualized field-programmable gate arrays. *IEEE Access* **2018**, *6*, 10497–10507. [CrossRef]
23. de Lima, O.A.; Costa, W.N.; Fresse, V.; Rousseau, F. A survey of NoC evaluation platforms on FPGAs. In Proceedings of the International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016.
24. Kidane, H.L.; Bourennane, E.B.; Ochoa-Ruiz, G. Run-time scalable noc for fpga based virtualized ips. In Proceedings of the IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Seoul, Korea, 18–20 September 2017; pp. 91–97.
25. Kidane, H.L.; Bourennane, E.B. MARTE and IP-XACT based approach for run-time scalable NoC. In Proceedings of the IEEE 12th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC), Hanoi, Vietnam, 12–14 September 2018; pp. 162–167.
26. Liu, Y.; Liu, P.; Jiang, Y.; Yang, M.; Wu, K.; Wang, W.; Yao, Q. Building a multi-FPGA-based emulation framework to support networks-on-chip design and verification. *Int. J. Electron.* **2010**, *97*, 1241–1262. [CrossRef]
27. Krishnaiah, G.; Silpa, B.V.; Panda, P.R.; Kumar, A. Fastfwd: An efficient hardware acceleration technique for trace-driven network-on-chip simulation. In Proceedings of the eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Scottsdale, AZ, USA, 24–29 October 2010; ACM: New York, NY, USA, 2010; pp. 247–256.
28. Wang, C.; Hu, W.H.; Lee, S.E.; Bagherzadeh, N. Area and power-efficient innovative network on-chip architecture. In Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), Pisa, Italy, 17–19 February 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 533–539.

29. Lotlikar, S.; Pai, V.; Gratz, P. AcENoCs: A configurable HW/SW platform for FPGA accelerated NoC emulation. In Proceedings of the 24th Internatioal Conference on VLSI Design (VLSI Design), Chennai, India, 2–7 January 2011; pp. 147–152.
30. Papamichael, M.K. Fast scalable FPGA-based network-on-chip simulation models. In Proceedings of the 2011 9th IEEE/ACM Internatioal Conference on Formal Methods and Models for Codesign (MEMOCODE), Cambridge, UK, 11–13 July 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 77–82.
31. Papamichael, M.K.; Hoe, J.C.; Mutlu, O. Fist: A fast, lightweight, FPGA-friendly packet latency estimator for noc modeling in full-system simulations. In Proceedings of the 2011 Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS), Pittsburgh, PA, USA, 1–4 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 137–144.
32. Ku, W.-C.; Chen, T.-F. Accelerating manycore simulation by efficient noc interconnection partition on FPGA simulation platform. In Proceedings of the 2011 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 25–28 April 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–4.
33. Pellauer, M.; Adler, M.; Kinsy, M.; Parashar, A.; Emer, J. Hasim: FPGA-based high-detail multicore simulation using time-division multiplexing. In Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture (HPCA), San Antonio, TX, USA, 12–16 February 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 406–417.
34. Tan, J.; Fresse, V.; Rousseau, F. Generation of emulation platforms for NoC exploration on FPGA. In Proceedings of the 22nd IEEE International Symposium on Rapid System Prototyping (RSP), Karlsruhe, Germany, 24–27 May 2011; pp. 186–192.
35. Heck, G.; Guazzelli, R.; Moraes, F.; Calazans, N.; Soares, R. HardNoC: A platform to validate networks on chip through FPGA prototyping. In Proceedings of the VIII Southern Conference on Programmable Logic (SPL), Bento Goncalves, Spain, 20–23 March 2012; pp. 1–6.
36. Fresse, V.; Ge, Z.; Tan, J.; Rousseau, F. Case study: Deployment of the 2d noc on 3d for the generation of large emulation platforms. In Proceedings of the 2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP), Tampere, Finland, 11–12 October 2012; pp. 23–29.
37. van Chu, T.; Sato, S.; Kise, K. Knocemu: High speed fpga emulator for kilo-node scale nocs. Embedded Multicore/Manycore SoCs (MCSoc). In Proceedings of the 2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs, Aizu-Wakamatsu, Japan, 23–25 September 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 215–222.
38. de Lima, O.A.; Fresse, V.; Rousseau, F. Evaluation of snmp-like protocol to manage a NOC emulation platform. In Proceedings of the 2014 International Conference on Field-Programmable Technology (FPT), Shanghai, China, 10–12 December2014; pp. 199–206.
39. Kang, J.M.; Bannazadeh, H.; Leon-Garcia, A. Savi testbed: Control and management of converged virtual ict resources. In Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, Ghent, Belgium, 27–31 May 2013.
40. Amazon Web Services EC2. FPGA Hardware and Software Development Kit. Available online: https://github.com/aws/aws-fpga (accessed on 2 December 2019).
41. Weerasinghe, J.; Abel, F.; Hagleitner, C.; Herkersdorf, A. Enabling FPGAs in hyperscale data centers. In Proceedings of the 15th IEEE UIC-ATC-ScalCom, Beijing, China, 10–14 August 2015.
42. Alveo Nimbix Cloud. Available online: https://www.nimbix.net/alveotrial (accessed on 23 March 2020).
43. Voss, N.; Quintana, P.; Mencer, O.; Luk, W.; Gaydadjiev, G. Memory mapping for multi-die FPGAs. In Proceedings of the IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), San Diego, CA, USA, 28 April–1 May 2019.
44. Voss, N.; Girdlestone, S.; Becker, T.; Mencer, O.; Luk, W.; Gaydadjiev, G. Low area overhead custom buffering for FFT. In Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 9–11 December 2019.
45. Li, X.; Maskell, D.L. Time-multiplexed FPGA overlay architectures: A survey. *ACM Trans. Des. Autom. Electron. Syst.* **2019**, *24*, 54. [CrossRef]
46. Jain, A.K. Architecture Centric Coarse-Grained FPGA Overlays. Ph.D. Thesis, Nanyang Technological University, Singapore, 2017.

47. Cheah, H.Y.; Fahmy, S.A.; Maskell, D.L. iDEA: A DSP block-based FPGA Soft Processor. In Proceedings of the 2012 International Conference on Field-Programmable Technology (FPT), Seoul, Korea, 10–12 December 2012.
48. Xilinx MicroBlaze Soft Processor Core. Available online: https://www.xilinx.com/products/design-tools/microblaze.html (accessed on 3 December 2019).
49. Altera Nios II Processor. Available online: https://www.altera.com/products/processors/overview.html (accessed on 4 December 2019).
50. Severance, A.; Lemieux, G.G.F. Embedded supercomputing in FPGAs with the VectorBlox MXP matrix processor. In Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, Montreal, QC, Canada, 29 September–4 October 2013.
51. Severance, A.; Lemieux, G. VENICE: A compact vector processor for FPGA Applications. In Proceedings of the 2011 IEEE Hot Chips 23 Symposium (HCS), Stanford, CA, USA, 17–19 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–5.
52. Chou, C.H.; Severance, A.; Brant, A.D.; Liu, Z.; Sant, S.; Lemieux, G.G. VEGAS: Soft vector processor with scratchpad memory. In Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays 2011, Monterey, CA, USA, 27 February–1 March 2011.
53. Yiannacouras, P.; Steffan, J.G.; Rose, J. VESPA: Portable, scalable, and flexible FPGA-based vector processors. In Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems, Atlanta, GA, USA, 19–24 October 2008.
54. Yu, J.; Lemieux, G.; Eagleston, C. Vector Processing as a Soft-core CPU Accelerator. In Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, 24–26 February 2008.
55. Cong, J.; Huang, H.; Ma, C.; Xiao, B.; Zhou, P. A fully pipelined and dynamically composable architecture of CGRA. In Proceedings of the 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, Boston, MA, USA, 11–13 May 2014.
56. Shukla, S.; Bergmann, N.W.; Becker, J. QUKU: A FPGA based flexible coarse grain architecture design paradigm using process networks. In Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium, Rome, Italy, 26–30 March 2007.
57. Landy, A.; Stitt, G. A low-overhead interconnect architecture for virtual reconfigurable fabrics. In Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems, Tampere, Finland, 7–12 October 2012.
58. Coole, J.; Stitt, G. Fast, flexible high-level synthesis from OpenCL using reconfiguration contexts. *IEEE Micro* **2014**, *34*, 42–53. [CrossRef]
59. Govindaraju, V.; Ho, C.H.; Nowatzki, T.; Chhugani, J.; Satish, N.; Sankaralingam, K.; Kim, C. DySER: Unifying functionality and parallelism specialization for energy-efficient computing. *IEEE Micro* **2012**, *32*, 38–51. [CrossRef]
60. So, H.K.-H.; Liu, C. FPGA Overlays. In *FPGAs for Software Programmers*; Springer: Cham, Germany, 2016.
61. Brant, A.D. Coarse and Fine Grain Programmable Overlay Architectures for FPGAs. MSc Thesis, University of British Columbia, Vancouver, BC, Canada, 2012.
62. Rashid, R.; Steffan, J.G.; Betz, V. Comparing performance, productivity and scalability of the TILT overlay processor to OpenCL HLS. In Proceedings of the 2014 International Conference on Field-Programmable Technology (FPT), Shanghai, China, 10–12 December 2014.
63. Paul, K.; Dash, C.; Moghaddam, M.S. reMORPH: A runtime reconfigurable architecture. In Proceedings of the 2012 15th Euromicro Conference on Digital System Design, Cesme, Izmir, Turkey, 5–8 September 2012.
64. Kapre, N.; Mehta, N.; Delorimier, M.; Rubin, R.; Barnor, H.; Wilson, M.J.; Wrighton, M.; DeHon, A. Packet switched vs. time multiplexed FPGA overlay networks. In Proceedings of the 2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 24–26 April 2006.
65. Papamichael, M.K.; Hoe, J.C. CONNECT: Re-examining conventional wisdom for designing nocs in the context of FPGAs. In Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays 2012, Monterey, CA, USA, 22–24 February 2012.
66. Huan, Y.; DeHon, A. FPGA optimized packet-switched NoC using split and merge primitives. In Proceedings of the 2012 International Conference on Field-Programmable Technology, Seoul, Korea, 10–12 December 2012.

67. Kapre, N.; Gray, J. Hoplite: Building austere overlay NoCs for FPGAs. In Proceedings of the 2015 25th International Conference on Field Programmable Logic and Applications (FPL), London, UK, 2–4 September 2015.
68. Brant, A.; Lemieux, G.G.F. ZUMA: An open FPGA overlay architecture. In Proceedings of the 2012 IEEE 20th international symposium on field-programmable custom computing machines, Toronto, ON, Canada, 29 April–1 May 2012.
69. Ferreira, R.; Vendramini, J.G.; Mucida, L.; Pereira, M.M.; Carro, L. An FPGA-based heterogeneous coarse-grained dynamically reconfigurable architecture. In Proceedings of the 14th International Conference on Compilers, Architectures and Synthesis for Embedded Systems, Taipei, Taiwan, 9–14 October 2011.
70. Kinsy, M.A.; Pellauer, M.; Devadas, S. Heracles: Fully synthesizable parameterized mips based multicore system. In Proceedings of the 21st International Conference on Field Programmable Logic and Applications, Chania, Greece, 5–7 September 2011.
71. Liu, C.; Yu, C.L.; So, H.K. A soft coarse-grained reconfigurable array based high-level synthesis methodology: Promoting design productivity and exploring extreme FPGA frequency. In Proceedings of the IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines, Seattle, WA, USA, 28–30 April 2013.
72. Gray, J. GRVI-phalanx: A massively parallel RISC-V FPGA accelerator. In Proceedings of the IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines, Washington, DC, USA, 1–3 May 2016.
73. Li, X.; Jain, A.; Maskell, D.; Fahmy, S.A. An area-efficient FPGA overlay using DSP block based time-multiplexed functional units. In Proceedings of the 2nd International Workshop on Overlay Architectures for FPGAs, Monterey, CA, USA, 21–23 February 2016.
74. Kumar, H.B.C.; Ravi, P.; Modi, G.; Kapre, N. 120-core microAptiv MIPS overlay for the Terasic DE5-NET FPGA board. In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 22–24 February 2017.
75. Byma, S.; Steffan, J.G.; Bannazadeh, H.; Garcia, A.L.; Chow, P. FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack. In Proceedings of the 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, Boston, MA, USA, 11–13 May 2014.
76. Fahmy, S.A.; Vipin, K.; Shreejith, S. Virtualized FPGA accelerators for efficient cloud computing. In Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, BC, Canada, 30 November–3 December 2015.
77. Tarafdar, N.; Lin, T.; Fukuda, E.; Bannazadeh, H.; Leon-Garcia, A.; Chow, P. Enabling flexible network FPGA clusters in a heterogeneous cloud data center. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17), Monterey, CA, USA, 22–24 February 2017; ACM: New York, NY, USA, 2017.
78. Knodel, O.; Genssler, P.R.; Spallek, R.G. Virtualizing reconfigurable hardware to provide scalability in cloud architectures. In Proceedings of the Tenth International Conference on Advances in Circuits, Electronics and Micro-electronics (CENICS 2017), Rome, Italy, 10–14 September 2017; IARIA: Wilmington, DE, USA, 2017.
79. 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, Mumbai, India, 2 September 2017.
80. Abbani, N.; Ali, A.; Doa'A, A.O.; Jomaa, M.; Sharafeddine, M.; Artail, H.; Akkary, H.; Saghir, M.A.; Awad, M.; Hajj, H. A distributed reconfigurable active SSD platform for data intensive applications. In Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications, Banff, AB, Canada, 2–4 September 2011.
81. Wang, W. pvFPGA: Accessing an FPGA-based hardware accelerator in a paravirtualized environment. In Proceedings of the 2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS), Montreal, QC, Canada, 29 September–4 October 2013; IEEE: Piscataway, NJ, USA, 2013.
82. Xia, T.; Prévotet, J.C.; Nouvel, F. Hypervisor mechanisms to manage FPGA reconfigurable accelerators. In Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016.

83. Jain, A.K.; Pham, K.D.; Cui, J.; Fahmy, S.A.; Maskell, D.L. Virtualized execution and management of hardware tasks on a hybrid ARM-FPGA platform. *J. Signal Process. Syst.* **2014**, *77*, 61–76. [CrossRef]
84. 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, Cagliari, Italy, 20–22 May 2014.
85. Bobda, C.; Majer, A.; Ahmadinia, A.; Haller, T.; Linarth, A.; Teich, J. The erlangen slot machine: Increasing flexibility in FPGA-based reconfigurable platforms. In Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology, Singapore, 11–14 December 2005.
86. Zhao, Q. Enabling FPGA-as-a-service in the cloud with hCODE platform. *IEICE Trans. Inf. Syst.* **2018**, *101*, 335–343. [CrossRef]
87. Weerasinghe, J.; Polig, R.; Abel, F.; Hagleitner, C. Network-attached FPGAs for data center applications. In Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT), Xi'an, China, 7–9 December 2016.
88. Tarafdar, N.; Eskandari, N.; Lin, T.; Chow, P. Designing for FPGAs in the Cloud. *IEEE Des. Test* **2017**, *35*, 23–29. [CrossRef]
89. 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. [CrossRef]
90. Pahl, C. Containerization and the PaaS cloud. *IEEE Cloud Comput.* **2015**, *2*, 24–31. [CrossRef]
91. Kirchgessner, R.; Stitt, G.; George, A.; Lam, H. 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), ACM, Monterey, CA, USA, 22–24 February 2012; pp. 205–208.
92. Asghari, M.; Rajabzadeh, A.; Dashtbani, M. 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), Mashhad, Iran, 20–21 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 72–77.
93. Najem, M.; Bollengier, T.; le Lann, J.C.; Lagadec, L. Extended overlay architectures for heterogeneous FPGA cluster management. *J. Syst. Archit.* **2017**, *78*, 1–14. [CrossRef]
94. Eskandari, N.; Tarafdar, N.; Ly-Ma, D.; Chow, P. A modular heterogeneous stack for deploying FPGAs and CPUs in the data center. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 24–26 February 2019; ACM: New York, NY, USA, 2019.
95. Vaishnav, A.; Pham, K.D.; Manev, K.; Koch, D. The FOS (FPGA Operating System) Demo. In Proceedings of the 29th International Conference on Field Programmable Logic and Application (FPL), Barcelona, Spain, 8–12 September 2019.
96. Intel OPAE. Available online: http://01.org/OPAE (accessed on 24 March 2020).
97. 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 (FPL), Ghent, Belgium, 4–8 September 2017.
98. Happe, M.; Traber, A.; Keller, A. Preemptive hardware multitasking in ReconOS. In *Applied Reconfigurable Computing*; Springer: Cham, Switzerland, 2015.
99. Rupnow, K.; Fu, W.; Compton, K. Block, drop or roll(back): Alternative preemption methods for RH multi-tasking. In Proceedings of the 17th IEEE Symposium on Field Programmable Custom Computing Machines, Napa, CA, USA, 5–7 April 2009.
100. Bourge, A.; Muller, O.; Rousseau, F. Generating efficient context-switch capable circuits through autonomous design flow. *ACM TRETS* **2016**, *10*, 1–23. [CrossRef]
101. Lubbers, E.; Platzner, M. ReconOS: An RTOS supporting hard-and software threads. In Proceedings of the 2007 International Conference on Field Programmable Logic and Applications, Amsterdam, Netherlands, 27–29 August 2007.
102. Peck, W.; Anderson, E.; Agron, J.; Stevens, J.; Baijot, F.; Andrews, D. Hthreads: A computational model for reconfigurable devices. In Proceedings of the 2006 International Conference on Field Programmable Logic and Applications, Madrid, Spain, 28–30 August 2006.
103. Shan, Y.; Wang, B.; Yan, J.; Wang, Y.; Xu, N.; Yang, H. FPMR: MapReduce framework on FPGA. In Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays 2010, Monterey, CA, USA, 21–23 February 2010.

104. Chen, Y.T.; Cong, J.; Fang, Z.; Lei, J.; Wei, P. When spark meets FPGAs: A case study for next generation DNA sequencing acceleration. In Proceedings of the 24th FCCM, Washington, DC, USA, 1–3 May 2016.
105. Mavridis, S.; Pavlidakis, M.; Stamoulias, I.; Kozanitis, C.; Chrysos, N.; Kachris, C.; Soudris, D.; Bilas, A. VineTalk: Simplifying software access and sharing of FPGAs in datacenters. In Proceedings of the 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, Belgium, 4–8 September 2017.
106. Eguro, K. SIRC: An extensible reconfigurable computing communication API. In Proceedings of the 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, Charlotte, NC, USA, 2–4 May 2010.
107. Intel FPGA. SDK for OpenCL. Programming Guide. UG-OCL002. 2016. Available online: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/opencl-sdk/archives/ug-aocl-programming-guide-16.1.pdf (accessed on 23 March 2020).
108. Segal, O.; Colangelo, P.; Nasiri, N.; Qian, Z.; Margala, M. SparkCL: A unified programming framework for accelerators on heterogeneous clusters. *arXiv* **2015**, arXiv:1505.01120.
109. Kulp, J.; Siegel, S.; Miller, J. *Open Component Portability Infrastructure (OPENCPI)*; Technial Report; Mercury Federal Systems Inc.: Arlington, VA, USA, 2013.
110. Intel HLS Compiler: Fast Design, Coding and Hardware. Available online: https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/hls-compiler.html (accessed on 24 March 2020).
111. Vivado High-Level Synthesis: Accelerates IP Creation by Enabling C, C++ and System C Specifications. Available online: https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html (accessed on 24 March 2020).
112. Open Spatial Programming Language (OpenSPL), Maxeler Technologies. Available online: https://www.maxeler.com/openspl-announced/ (accessed on 24 March 2020).
113. Pell, O.; Mencer, O.; Tsoi, K.H.; Luk, W. Maximum performance computing with dataflow engines. In *Computing in Science & Engineering*; IEEE: Piscataway, NJ, USA, 2012; pp. 98–103.
114. Fleming, K.; Adler, M. The LEAP FPGA Operating System. In *FPGAs for Software Programmers*; Springer: Cham, Switzerland, 2016.
115. Vesper, M.; Koch, D.; Vipin, K.; Fahmy, S.A. JetStream: An open-source high-performance PCI express 3 streaming library for FPGA-to-Host and FPGA-to-FPGA communication. In Proceedings of the 2016 26th international conference on field programmable logic and applications (FPL), Lausanne, Switzerland, 29 August–2 September 2016.
116. Jacobsen, M.; Richmond, D.; Hogains, M.; Kastner, R. RIFFA 2.1: A reusable integration framework for FPGA accelerators. *ACM TRETS* **2015**, *8*, 1–23. [CrossRef]
117. Yoshimi, M.; Nishikawa, Y.; Miki, M.; Hiroyasu, T.; Amano, H.; Mencer, O. A performance evaluation of CUBE: One Dimensional 512 FPGA cluster. In Proceedings of the International Symposium on Applied Reconfigurable Computing, ARC 2010, Bangkok, Thailand, 17–19 March 2010; Springer: Heidelberg, Germany, 2010.
118. Wang, Z.; Zhang, S.; He, B.; Zhang, W. Melia: A MapReduce framework on OpenCL-Based FPGAs. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3547–3560. [CrossRef]
119. Yeung, J.H.C. Map-reduce as a programming model for custom computing machines. In Proceedings of the 2008 16th International Symposium on Field-Programmable Custom Computing Machines, Palo Alto, CA, USA, 14–15 April 2008.
120. Tsoi, K.H.; Luk, W. Axel: A Heterogeneous Cluster with FPGAs and GPUs. In Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays 2010, Monterey, CA, USA, 21–23 February 2010.
121. Iordache, A.; Pierre, G.; Sanders, P.; de FCoutinho, J.G.; Stillwell, M. High performance in the cloud with FPGA groups. In Proceedings of the 9th International Conference on Utility and Cloud Computing, Shanghai, China, 6–9 December 2016.
122. Huang, M. Programming and runtime support to blaze FPGA accelerator deployment at datacenter scale. In Proceedings of the Seventh ACM Symposium on Cloud Computing, SoCC '16, Santa Clara, CA, USA, 5–7 October 2016.

123. Theodoropoulos, D.; Alachiotis, N.; Pnevmatikatos, D. Multi-FPGA evaluation platform for disaggregated computing. In Proceedings of the 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA, USA, 30 April–2 May 2017.
124. Ouyang, J.; Lin, S.; Qi, W.; Wang, Y.; Yu, B.; Jiang, S. SDA: Software-defined accelerator for large-scale DNN systems. In Proceedings of the IEEE Hot Chips 26 Symposium (HCS), Cupertino, CA, USA, 10–12 August 2014.
125. El-Araby, E.; Gonzalez, I.; El-Ghazawi, T. Virtualizing and sharing reconfigurable resources in high-performance reconfigurable computing systems. In Proceedings of the 2008 Second International Workshop on High-Performance Reconfigurable Computing Technology and Applications, Austin, TX, USA, 17 November 2008.
126. Zheng, Z.; Wang, T.; Weng, J.; Mumtaz, S.; Bashir, A.K.; Hussain, C.S. Differentially private high-dimensional data publication in internet of things. *IEEE Internet Things J.* **2019**. [CrossRef]
127. Bashir, A.K.; Ohsita, Y.; Murata, M. A distributed virtual data center network architecture for the future internet. *IEICE Tech. Rep. (IN2014–165)* **2015**, *114*, 261–266.
128. Bashir, A.K.; Ohsita, Y.; Murata, M. Abstraction layer based virtual data center architecture for network function chaining. In Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW)—ICDCS, Nara, Japan, 27–30 June 2016.
129. Bashir, A.K.; Ohsita, Y.; Murata, M. Abstraction layer based distributed architecture for virtualized data centers. In Proceedings of the Cloud Computing 2015: 6th International Conference on Cloud Computing, Grids, and Virtualization, Nice, France, 22–27 March 2015; pp. 46–51.
130. Flynn, M.J. Flynn's taxonomy. *Encycl. Parallel Comput.* **2011**, 689–697. [CrossRef]
131. Caldeira, P.; Penha, J.C.; Bragança, L.; Ferreira, R.; Nacif, J.A.; Ferreira, R.; Pereira, F.M. From Java to FPGA: An experience with the intel HARP system. In Proceedings of the 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Lyon, France, 24–27 September 2018.
132. Feng, L.; Zhao, J.; Liang, T.; Sinha, S.; Zhang, W. LAMA: Link-Aware Hybrid Management for Memory Accesses in Emerging CPU-FPGA Platforms. In Proceedings of the 56th ACM/IEEE Design Automation Conference, Las Vegas, NV, USA, 2–6 June 2019.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).