Received 24 May 2023, accepted 15 June 2023, date of publication 21 June 2023, date of current version 10 July 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3288431*

SURVEY

# A Survey on FPGA-Based Heterogeneous Clusters Architectures

**WERNER FLORIAN SAMAYOA** [1,2], **MARIA LIZ CRESPO** [1], **ANDRES CICUTTIN** [1], **AND SERGIO CARRATO** [2]

[1]Multidisciplinary Laboratory (MLab), The Abdus Salam International Centre for Theoretical Physics, 34151 Trieste, Italy
[2]Dipartimento di Ingegneria e Architettura (DIA), Universitã degli Studi di Trieste, 34127 Trieste, Italy

Corresponding author: Werner Florian Samayoa (werneroswaldo.floriansamayoa@phd.units.it)

This work was supported by the University of Trieste and The Abdus Salam International Centre for Theoretical Physics.

**ABSTRACT** In recent years, the most powerful supercomputers have already reached megawatt power consumption levels, an important issue that challenges sustainability and shows the impossibility of maintaining this trend. To this date, the prevalent approach to supercomputing is dominated by CPUs and GPUs. Given their fixed architectures with generic instruction sets, they have been favored with lots of tools and mature workflows which led to mass adoption and further growth. However, reconfigurable hardware such as FPGAs has repeatedly proven that it offers substantial advantages over this supercomputing approach concerning performance and power consumption. In this survey, we review the most relevant works that advanced the field of heterogeneous supercomputing using FPGAs focusing on their architectural characteristics. Each work was divided into three main parts: network, hardware, and software tools. All implementations face challenges that involve all three parts. These dependencies result in compromises that designers must take into account. The advantages and limitations of each approach are discussed and compared in detail. The classification and study of the architectures illustrate the trade-offs of the solutions and help identify open problems and research lines.

**INDEX TERMS** FPGA, SoC, heterogeneous computing, supercomputing, reconfigurable computing.

The notion of reconfigurable hardware has been present since 1984, when Altera delivered the first programmable logic device (PLD) to the industry [1]. Then, in 1985 Ross Freeman and Bernard Vonderschmitt patented the first commercially viable field-programmable gate array (FPGA) [2]. Owing to production costs, when compared to application-specific integrated circuits (ASICs), FPGAs are traditionally used in applications with low production volumes that require high throughput and low latency.

FPGAs are electronic devices that consist of many configurable logic blocks composed of look-up tables, flip-flops, I/O blocks, and interconnection fabric. FPGAs are used to create custom hardware solutions, which make the implementation of algorithms quite different from targeting a CPU. The initial step typically consists of describing the algorithm using a Hardware Description Language (HDL), such as VHDL or Verilog. The HDL description is then synthesized into a netlist that is mapped onto the FPGA's logic elements and interconnections required to implement the desired digital design. The final implementation in the FPGA is performed using vendor-specific tools such as Vivado [3], Vitis [4], Quartus [5], and Libero [6]. Once the mapping and routing process is completed, the design is compiled into a bitstream file loaded onto the FPGA to configure its logic elements and interconnections to create a circuit corresponding to the algorithm. It has to be added, however, proprietary FPGA vendor tools have dominated the field, there are now some open-source FPGA tools, such as Yosys [7], F4PGA [8], and RapidSilicon [9], that provide alternative options for developers seeking open-source solutions.

FPGAs have evolved into more complex devices [10] by integrating components, such as embedded memory resources, clock management units, digital signal processing

The associate editor coordinating the review of this manuscript and approving it for publication was Vincenzo Conti.

**TABLE 1. The 13 dwarfs of Berkeley [16], where each one represents an algorithmic method encapsulating patterns of communication and/or computation with example problems.**

| | Dwarf | Examples/Applications |
|---|---|---|
| 1 | Dense Matrix | Linear algebra (eg. Cholesky decomposition) |
| 2 | Sparse Matrix | Linear algebra (eg. Machine learning) |
| 3 | Spectral | FFT-based methods |
| 4 | N-Body | Particle-particle interactions, molecular dynamics |
| 5 | Structured Grid | Fluid dynamics, meteorology |
| 6 | Unstructured Grid | Adaptive mesh FEM |
| 7 | MapReduce | Monte Carlo integration, distributed pattern-based searching |
| 8 | Combinational | Logic gates (e.g., Toffoli gates) |
| 9 | Graph traversal | Searching, selection |
| 10 | Dynamic Programming | Tower of Hanoi problem |
| 11 | Backtrack/Global optimization | Branch-and-Bound |
| 12 | Graphical Models | Probabilistic networks |
| 13 | Finite State Machine | Transistor-transistor logic counter |

blocks (DSP), network-on-chip (NoC), and CPUs [11]. These hybrid devices are known as system-on-chip (SoC-FPGA) or adaptive SoCs, depending on the vendor. Their increased capabilities have increased interest in specific applications and general purposes [12], [13].

As a reconfigurable device, FPGA offers the advantage of continuous improvement in hardware and software. In fact, being able to change the architecture offers great freedom when developing complex systems. Furthermore, FPGAs have been shown to consume considerably less power than CPUs and GPUs [14], leading to reduced cooling and energy costs.

By studying computing problems, classification based on repeating algorithmic patterns was proposed in 2004 [15]. In 2006, [16] 6 new algorithmic encapsulations were defined, expanding the classification to 13 dwarfs as shown in Table 1. Theoretically, each dwarf can be mapped onto a specific computing architecture [17], [18]. This has inspired the creation of benchmarks for heterogeneous systems such as DwarfBench [19], Rodinia [20], and OpenDwarfs [21].

Several implementations of heterogeneous high-performance computing (HPC) systems housing FPGAs can be named, such as *Project Catapult* at Microsoft [22], *Alibaba FaaS* (FPGA as a Service) [23], *Amazon EC2 F1* instances [24], and *ARUZ* cluster at Lodz University [25]. At CERN, the massive adoption of FPGAs for online data processing has motivated the development and adoption of specific tools to aid the development of applications based on FPGAs, such as hls4ml [26] (high-level synthesis for machine learning). This tool, along with many others [27], [28], [29], [30], [31], allows for a higher level of abstraction, thereby significantly reducing implementation errors and development time. The preference for FPGAs is due to their reconfigurability, which allows extreme hardware specialization when needed. In addition, the fact that FPGAs offer a wide array of input-output ports makes them ideal for stream computation and for creating pipe-lined systems that can maintain high throughput with low latency.

The purpose of this survey is to demonstrate and analyze the challenges of heterogeneous supercomputing by studying the most relevant implementations of FPGA-based cluster architectures from different application fields. Each studied platform provides valuable insight into the decisions and tradeoffs developers have made to reach their specific goals. By leveraging their experience, it will be possible to visualize the evolution and present trends in FPGA-based clusters and target the main open challenges. We propose dividing the architectural components of each cluster into network, hardware, and software tools. This division helps identify and discuss the pros and cons of each component in its corresponding domain.

The main contributions of this study are as follows:

1) The comprehensive study of the state-of-the-art of FPGA-based clusters.
2) A three-way segmentation of the clusters' architecture.
3) A critical discussion of the components that build up the studied clusters.

In the context of this paper, we describe a cluster by its computational units (CU), which correspond to its smallest independent part and sometimes coincide with a single network node. Each CU can be composed of several computational elements (CE), namely CPUs, GPUs, and FPGAs.

The remainder of this paper is organized as follows. Section I elaborates on the implementations and explores relevant advancements in their application fields. A table at the end of each application field discussion summarizes the main contributions of each study, along with the reported performance and energy improvements, when available. Significant differences can be understood by studying the evolution of heterogeneous clusters within each niche. Section II presents the classification of systems from an architectural perspective. The three main aspects described in each implementation were used as comparison points. Subsection II-A presents a comparison of the network infrastructure in the studies. The hardware available in each studied cluster is discussed in Subsection II-B. To complete the classification discussion, the developer tools are compared in Subsection II-C. In Section III we present the remaining open problems and the identified trends. To close this paper, Section IV draws conclusions.

## I. CLUSTER IMPLEMENTATIONS

Different FPGA-based cluster implementations were studied, and their specific characteristics highlight the purpose for which they were planned. Technological advances offer greater flexibility, and cost reduction opens the door to increasing complexity. It can be appreciated that there is a growing interest in developing research-capable platforms to explore diverse areas of heterogeneous supercomputing. Tables 2, 3, 4, 5, and 6 provide a summary of the contributions of each work and the reported energy and performance, if available.

### A. MANYCORE EMULATION

The development of manycore platforms is a long and expensive process that involves several stages of experimentation, validation, and integration. There are software tools that help simulate architectures for easy parameter tuning, with the major drawback of speed. In this particular aspect, FPGA prototyping allows faster execution times and benefits from insights from real hardware. It is not rare for a complete platform to exceed the logic available in a single FPGA, pushing for a cluster of FPGAs.

This was the case since 1997, when one of the first FPGA clusters was used to emulate the *RAW* architecture [32]. The RAW cluster consisted of 5 boards or CUs, each with 64 FPGAs, totaling 320 FPGAs. Its results showed orders of magnitude speed-up compared to contemporaneous scalable processors with the disadvantages of reduced flexibility, high cost, and high implementation complexity, which hindered their adoption in other research applications.

In 2006, the *FAST* [33] cluster was presented to bring hardware back into the research cycle to address the disadvantages of *RAW*. FAST combined dedicated microprocessor chips and static random access memories (SRAM) with FPGAs into a heterogeneous hybrid solution to simulate chip multiprocessor architectures. The vision was to reduce hardware costs and ease development, both for programming and portability. Each *FAST* CU consisted of 8 processors, 10 Xilinx Virtex FPGAs, and 4 memory-interconnected tiles. The 2 processors in each tile acted as the CPU and floating processing unit, respectively, and 2 FPGAs acted as the level-one memory controller and coprocessor.

A central hub, made up of 2 FPGAs, was used to manage shared resources and orchestrate communication between tiles allowing access to off-the-board devices through external IOs. Additionally, the expansion connector available to the FPGA hub allows multiple *FAST* CUs to be connected. The CU implementation is illustrated in Figure 1.

A custom software stack was developed specifically for *FAST*. It included several modules and predefined interfaces for functionality and benchmarking. An operating system was developed to manage control tasks such as programming and configuration. Portability was demonstrated by implementing several architectures; however, scalability and costs remained open to discussion.

Similar to *FAST*, the *RAPTOR* cluster was presented as a baseboard hosting up to 4 daughter cards based on complex programmable logic devices (CPLD) [34]. In 2010, a second version was presented using FPGAs and a renewed architecture [35]. This new version consisted of a *RAPTOR-Xpress* baseboard (CU) that provides two buses for Gigabit Ethernet, universal serial bus (USB) 2.0, and peripheral component

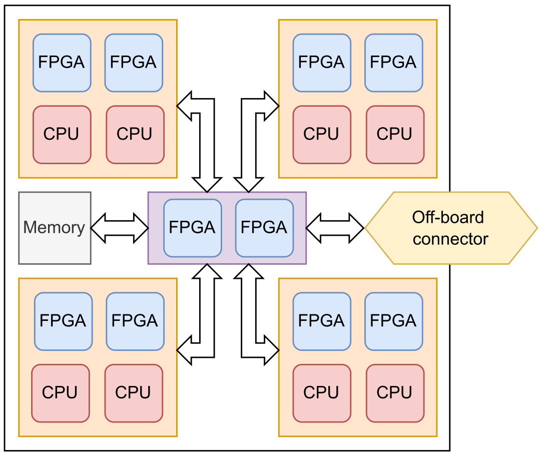


**FIGURE 1. Fast [33] computational unit (CU) with the computing tiles in orange and the FPGA hub in purple.**

interconnect express (PCIe) 2.0 × 8 for the host connection to configure and manage up to 4 DB-V5 (daughter board version 5).

Figure 2 shows the *RAPTOR-Xpress* baseboard with 4 DBs interfaced directly with their neighbors in a ring topology. Each has a Xilinx Virtex-5 FPGA with up to 4 GB of DDR3 memory and a dedicated FPGA as a PCIe interface Multiple baseboards can be connected together via 4 high-speed connectors, each consisting of 21 full-duplex serial lanes, enabling scaling resources beyond the 4 DB on board. The baseboards can also be interfaced with the host via dedicated FPGAs on Nallatech front-side bus acceleration modules [36], which provides an extra 8.5 Gb/s for writing and 5.6 Gb/s for reading.

The *RAPTOR* project also comprises a custom software development environment that includes RAPTORLIB, RAPBTORAPI, and RAPTORGUI tools, which aid developers by providing hardware-supported protocols, remote access, and a graphical user interface to facilitate testing. The design flow includes aids for design partitioning, which is a manual process assisted by a graphical integrated development environment (IDE) and standard synthesis tools developed in vMAGIC [37].

Convinced by the need for cheaper and smaller hardware, the *Formic* cluster [38] based on the *Formic* board [39] was presented in 2014. The *Formic* board acts as the building block for a larger system, with a maximum size of 4096 boards. Each board consists of an FPGA, SRAM, 1 GB of double data rate (DDR) RAM, a power supply, buffered joint test action group (JTAG) connectors, and configuration memory, making it independent and perfectly symmetric. Eight multi-gigabit transceivers (MGT) at a maximum speed of 3 Gb/s are available for interconnection on 8 serial advanced technology attachment (SATA) connectors. Inside each board, a full NoC with a 22 port crossbar switch interfaces the configured blocks with MGT links and allows developers to scale the designs. Access to local and remote memories is done using the Remote Direct Memory

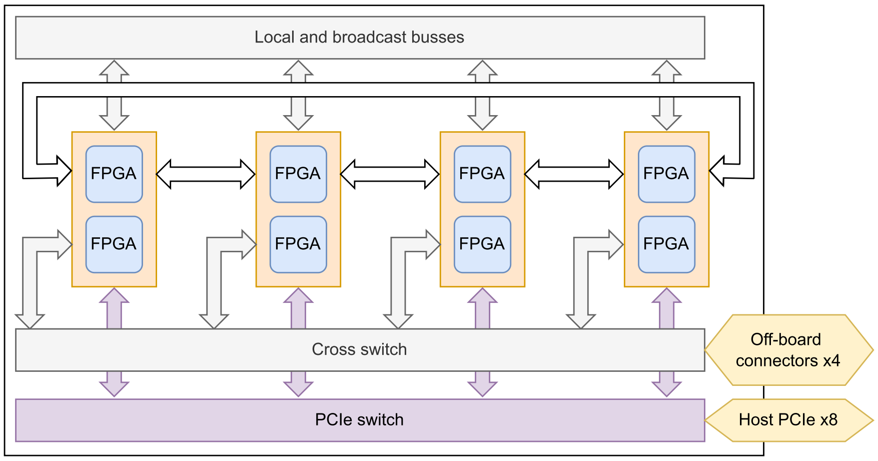


**FIGURE 2. Simplified diagram of the RAPTOR-Xpress board [35] or computational unit (CU) with the daughter boards in orange.**

Access (RDMA) protocol [40]. As the first application, a multicore system based on 8 custom MicroBlaze [41] processors per module forming a 512-core cluster [42] was implemented.

Simultaneously, the industry has produced exciting developments in manycore emulation. In an attempt to reduce the time to market for new ICs, *Cadence* [43] and *Siemens* [44], together with others, developed solutions for the prototyping of ICs. Unfortunately, there is little accessible information regarding the architecture of most implementations, and the high costs make them uncommon in academia, with some exceptions, such as the *Pico Computing* board (now *Micron*) used for image processing [45] and the *DINI* (now *Synopsys*) board FPGA board used for online video processing [46].

From the described works, it can be seen that there is a trend in reducing the complexity of CUs, as shown in Figure 3. In this field, costs tend to be the leading factor, making granularity a desirable characteristic. With smaller CUs, it is possible to reduce the implementation costs, depending on the requirements of the chip to emulate. Smaller CUs also make it easier for clusters to scale, maintain, and upgrade.

### B. SCIENTIFIC COMPUTING

The complexity of scientific computing problems has always pushed technology to its limit, making computer clusters a basic requirement. Regardless of whether complex algorithms process huge amounts of data or massive system simulations, reconfigurable computing provides the level of customization required by these problems. This did not go unnoticed, as early as 1991, programmable hardware was already part of custom supercomputers for specific problems like in *RTN* [47], *RASA* in 1994 [48], and later in *SUE* 2001 [49].

The first massive cluster was created in 2006. *Janus* [50] was a massively parallel modular cluster for the simulation of specific theoretical problems in physics developed by a large collaboration of European institutions [51].

The core of *Janus* comprised an array of 4 by 4 FPGA-based simulation processors (SP) which were connected with their nearest neighbors. Another processing unit called an Input/Output processor (IOP), acted as a crossbar and was in charge of managing communications between FPGAs and the host.

A two-layer software stack was created to help developers build applications. The firmware layer consisted of a fixed part targeting the IOPs, which included a stream router and dedicated devices to communicate, manage, and program the SPs. The second layer, the *Janus* Operating System (JOS), consisted of the programs running on the host PCs, including a set of libraries (JOSlib) to manage the IOP devices, a Unix socket application program interfaces (APIs) to integrate high-level applications and new SP modules, and an interactive shell (JOSH) for debugging and testing.

In the worst case, *Janus* performed just 2.4 times faster than conventional PCs. Nonetheless, *Janus* was limited by its performance and scarce memory for some applications [52].

In parallel, great interest has been shown in the cryptanalysis field with the development of the *COPACOBANA* FPGA cluster [53] in 2006. Figure 4 shows the *COPACOBANA* cluster which was built over a CU holding up to 20 dual in-line memory modules each with 6 Xilinx Spartan-3 FPGAs directly connected to a 64-bit data bus and 16-bit control bus. A controller module allowed the host PC to interact via USB or Ethernet through a software library that provided the necessary functions for the PC to program, store, and read the status of the cluster as a whole or as individual FPGAs. This made it possible to scale resources by attaching another CU to the host PC. Its capabilities were demonstrated by testing several encryption algorithms, which resulted in it outperforming conventional computers by orders of magnitude [54].

The positive outcome of this project motivated the creation of a hybrid FPGA-GPU cluster [55] based on commercial off-the-shelf (COTS) components in 2010. The *Cuteforce* [56] system implemented 15 CUs, 14 with Xilinx Virtex FPGAs,

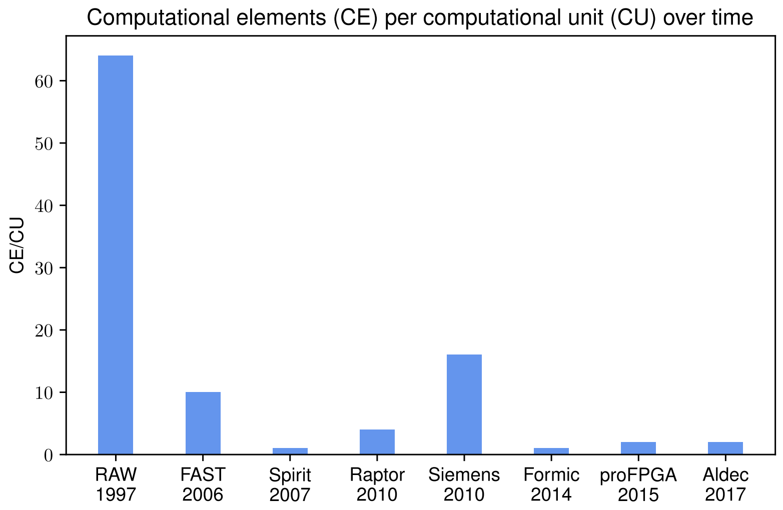


**FIGURE 3.** **Clusters targeting manycore emulation have shown a trend of reducing the complexity and increasing the granularity of CUs to favor production costs and scalability.**

**TABLE 2.** **Manycore emulation clusters' contributions and reported performance improvement.**

| Work | Contribution | Performance |
|---|---|---|
| RAW 1997 [32] | First academic FPGA cluster for the emulation of multicore systems. | 10 to 1000 times more performant than commercial Sparc 20/71. |
| FAST 2006 [33] | First hybrid platform for manycore research using dedicated FPGAs, SRAMs, and microprocessors. | Over 2 times more performant than any previous emulation platform. |
| RAPTOR 2010 [35] | Scalable MPSoC emulator with multiplexed PCIe access from host. | |
| Formic 2010 [39] | Cost-efficient scalable cluster based on a minimal independent building block. | 50000 times faster than that of the software simulation. |

and the last with an NVIDIA GPU interconnected through a CPU on a CU via Infiniband. The results were not as expected, partly because of complications in FPGA implementation.

The same approach was later used in 2010 by *Tse*, et al. [57] who focused on Monte Carlo simulations. However, instead of using one CE per CU, a single CU was used to host 2 CPUs, an NVIDIA GPU, and a Xilinx FPGA, which was further supported by a comprehensive analysis of the performance and energy. The network remained practically unchanged from *Cuteforce*, where the CPUs are the main communication CEs and relegate GPUs and FPGAs to an accelerator position. To further demonstrate the scalability of this strategy, *Superdragon* [58] was created to accelerate single-particle cryo-electron 3D microscopy.

*Bluehive* [59] also sought to distance itself from custom PCBs by embracing commodity boards to build a custom FPGA cluster for scientific simulations and manycore emulation [60] requiring high-bandwidth and low-latency communication. These challenges were overcome with the development of a 64-node FPGA cluster based on Terasic DE4 boards that host an Altera Stratix IV FPGA, an 8xPCIe connector, and a DIMM with 4 GB of RAM and interfaced through a custom interconnect called BlueLink [61] with four 8U rack boxes, each with 16 boards. The boards in the boxes were interconnected through a PCIe to the eSATA board. A small Linux computer allowed remote programming using a USB-to-JTAG converter and a DE2 board as a JTAG fan-out to parallelize the configuration.

The *Bluehive* development environment was supported by Quartus and mandatory blocks were provided to developers, routers for inter-FPGA communication, FBs, and high-speed serial link controllers [61], all developed on Bluespec SystemVerilog [62].

In 2014, *Janus* received an important upgrade [63], which significantly improved its performance. The architecture remained mostly the same, with the largest change in the adoption of newer FPGAs with 8 GB of RAM and MGTs instead of ordinary I/Os for interconnection.

*Janus II* and *Bluehive* were successful in tackling the memory issue, but as problems scale, larger clusters were needed. This was the case for *ARUZ* [25], an application-specific cluster formed by approximately 26,000 FPGAs distributed over 20 panels, each consisting of 12 rows, which in turn contained 12 CU. The CUs are composed of eight slave FPGAs that constitute the resources and a central master SoC-FPGA that manages operations. The addition of the Zynq SoC is motivated by the higher abstraction level provided by the

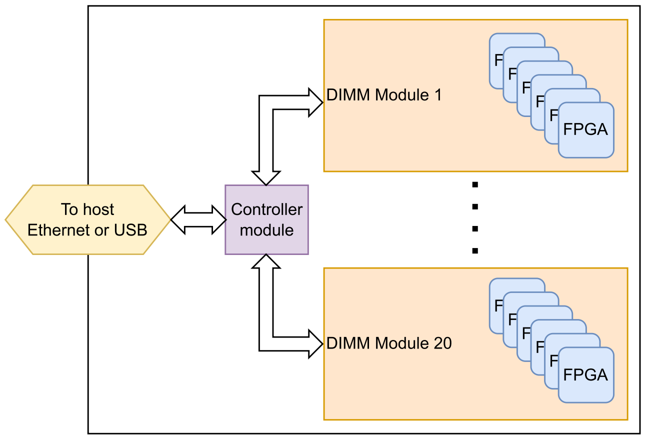


**FIGURE 4. COPACOBANA [53] computational unit (CU) with the dual in-line memory module (DIMM) modules in orange, each with 6 FPGAs, and the controller module in purple.**

ARM processor for slow-control tasks. In addition, each CU is interfaced with a concentrator board (CB) that feeds the state of the simulations to a host that controls the entire process.

Global communication is based on Gigabit Ethernet and allows data exchange between SoC-FPGAs to configure its 8 FPGAs. All nodes are connected in a Daisy chain, and only one board is connected to an external switch. A custom protocol for data transfer was developed, consisting of a small packet of no more than 256 bytes, with a constant overhead of 11 bytes.

*ARUZ* designers developed their own methodology [64], as there are no standard solutions available. Considering the multitude of mechanism combinations for programming and controlling *ARUZ*, a high level of flexibility is required. VPP [65] was selected for code pre-processing and parameterization. DLLDesigner was developed to generate VHDL code for interconnecting as many FBs as required. All of these tools allow the implementation of highly optimized architectures for molecular simulations.

FPGAs have also found a place in neuromorphic computing, as demonstrated by *Bluehive*. Spiking neural networks (SNN) require many densely interconnected elements. A substantial level of parallelism is suitable for hardware acceleration; however, the challenge is scalability. This was specifically addressed by *Astrobyte* [66] using a fully scalable NoC-based FPGA cluster with functional verification and real-time monitoring.

However, more specialized platforms presented better results at higher costs. This is the case for *BiCoSS* [67], a 35 system-on-module cluster, each with a Cyclone IV FPGA and 2 SDRAMs capable of simulating 4 million spiking neurons in real-time.

Another relevant application in the scientific context is real-time control (RTC) systems of adaptive optics (AO) instruments. This is the main focus of the *Green Flash* project [68] that aims to develop energy-efficient real-time HPC accelerators and smart interconnects, based on GPUs and FPGAs [69]. The RTC modules have a standard CPU server that hosts an NVIDIA GPU, Intel CPU, and Intel Arria 10 FPGA. The FPGAs are hosted on a custom mainboard called $\mu XComp$ which includes 2 GB of onboard RAM, PCIe 3, an FMC connector, Ethernet, 4 QSFP, and other valuable resources.

In this heterogeneous system, communication between GPUs is performed by a Smart Interconnect (SI) system implemented on FPGAs. The SI uses the UDP protocol, which is implemented in the FPGA fabric alongside the device protocol handlers and dedicated direct memory access (DMA) engines. This is configured with the QuickPlay FPGA framework, which extends its capabilities by using abstraction models and board support packages (BSPs) for portability. This architecture allows pipelining several GPUs and FPGAs. A similar approach can be seen in *Spinnaker* [70] and *BrainscaleS* [71] supercomputers, which implement dedicated ASICs interconnected by FPGAs for neuromorphic computing, and *MDGRAPE* [72] for modular dynamics simulations.

### C. FPGAS IN DATA CENTERS

The positive results obtained by FPGAs attracted great interest outside of the scientific community. Specifically in the data center (DC) context, where computing tasks can quickly overwhelm CPUs. DC workloads demand reduced power consumption, latency, and cost while maximizing computing power and flexibility.

*Catapult* [22] is a successful example of the inclusion of FPGAs in high-reliability commodity DC. FPGAs were specifically selected given that the flexibility of reconfigurable hardware helps tackle the 2 main requests in DCs. First, the desire for homogeneity greatly facilitates the installation, maintenance, and deployment of services. Second, there is a need for flexibility, considering that such services evolve rapidly, making fixed hardware impractical.

A custom half-width unit motherboard was developed to host 2 high-end CPUs and the daughter FPGA card, which consisted of a Stratix V D5 FPGA with 8 GB of DRAM and acted as the CU. Two 12-core Sandy Bridge processors with 64 GB of RAM, 2 SSDs, and 4 HDDs complete the resources present on the motherboard. The FPGA and host CPUs communicate via PCIe, and high-speed transceivers are used in the inter-FPGA network. A two-dimensional $6 \times 8$ node torus was selected for the network configuration in each rack. For the final system, 34 of these racks were used for a total of 1632 nodes.

To evaluate the performance of *Catapult*, a significant portion of the ranking stack of Bing was offloaded to each rack. To guarantee the reliability of the system, the following services were implemented:2-bit error detection and 1-bit error correction on top of the CRC in the DRAM and high-speed network. For user productivity and reusability, the FPGA space was split into 2 parts. A shell that hosts hardware controllers, an inter-FPGA network stack, a status notifier,

**TABLE 3.** Scientific computing clusters' contributions, reported power and performance gains.

| Work | Contribution | Power | Performance |
|---|---|---|---|
| Janus 2006 [50] | Fine-grained memory structures for statistical physics' simulations on a reconfigurable cluster, evidence of memory as the performance bottleneck. | 9 times more efficient than a Core 2 Duo. | 1000 times faster than Intel Core 2 Duo 2.0 GHz. |
| COPACOBANA 2006 [53] | Cost effective FPGA cluster for cryptanalysis. | 8 times more efficient than a PC cluster. | more than 650 times faster than an equivalent cost PC (Pentium-M) for exhaustive DES key search. |
| Cuteforce 2010 [56] | Efficient framework for FPGA-GPU collaborative cluster programming | | 7.9 million keys per second for PDF encryption brute force attack |
| Tse, et al. 2010 [57] | Research on efficiency of collaborative clusters of CPUs, GPUs, and FPGAs with dynamic scheduling showing that FPGA and GPU collaboration offers the best energy-performance trade-off. | 19.6 times more efficient. | 44 times faster with FPGA-GPU nodes over 2 CPU nodes (AMD Phenom 9650 quad-core). |
| Bluehive 2012 [59] | Usage of commercially available boards as building blocks for simulating millions of neurons in real-time | | 162 times faster than software simulation |
| Janus II 2014 [63] | Janus update with more memory, bigger FPGAs, and tightly coupled to host computers. | 12 times more efficient than Xeon-Phi. | 26 times faster than a Xeon-Phi. |
| Superdragon 2015 [58] | CPU, GPU, and FPGA cluster for 3D reconstruction of Cryo-electron microscopy mixing parallel patterns. | GPU and FPGA improved efficiency by 7.2 and 14.2 respectively compared to a Multicore CPU. | GPU and FPGA speed up of 8.4 and 2.25 respectively compared to a Multicore CPU. |
| ARUZ 2018 [25] | Biggest FPGA cluster to date based on the Dynamic Lattice Liquid model with low latency communication. | 1.6 times more efficient than a 6-core CPU for simulating 1.5 million molecules | 1600 times faster than a 6-core CPU. |
| Green Flash 2018 [69] | Heterogeneous cluster for critical real-time application.s | FPGA and GPU parts show double the energy efficiency when compared to Xeon Phi CPUs | GPU and FPGA parts show an improvement of 2.6 and 3.5 respectively over Xeon Phi CPUs. |
| Astrobyte 2020 [66] | FPGA cluster for simulating spiking neuron networks with NoC routing mechanisms. | | 188 times faster compared to Matlab implementation. |
| BiCoSS 2021 [67] | Biologically inspired multi-FPGA cluster for neuromorphic computing. | 2.8 thousand times more efficient than an NVIDIA GTX 280 GPU platform. | Computational efficiency of $5.4 \times 10^5$ times more than traditional CPU-based serial solutions. |

and a single-event upset logic to reduce system errors that consume 23% of the FPGA resources, and a role part where the computing logic lies. Additionally, a Mapping manager and health monitor continually scanned each node in the network. In case of failure, the faulty node is immediately reconfigured. If the issue persists, the node is flagged for manual intervention, and the mapping manager automatically relocates the services to the available resources.

With custom hardware and communication protocol, Catapult achieved an improvement of 95% in throughput in a production search infrastructure when compared to a software-only solution. In addition, the inclusion of the FPGA increased the power consumption by only 10%, and the added cost of ownership did not exceed the limit of 30%. These results show the significant advantage that FPGAs can offer in terms of throughput and power consumption.

With the success of *Catapult* [22], it was only a matter of time before FPGAs were made available for cloud computing tasks, which is exactly what the *IBM cloudFPGA* [73] did. Virtualizing the user space makes FPGAs in an Infrastructure-as-a-Service (IaaS) environment feasible for education, research, and testing.

In the architecture presented, the FPGAs are standalone nodes in the cluster directly interfaced to the DC via PCIe, unlike the approach of *Amazon* [24], *Alibaba* [23] and *IBM*

*Supervessel* [74] which tie the FPGAs to host CPUs. Under this approach, a daughter card consisting of an FPGA and abundant RAM was developed. By creating a custom carrier board, 64 daughter cards can be accommodated in a single 2U rack chassis [75]. To achieve the desired homogeneity within the DC, FPGAs have been provided with a soft network interface chip, with the advantage of loading only the required services.

The multi-FPGA fabric formed by multiple prototypes of a network-attached FPGA was evaluated with a text-analytics application. The results, compared to a software implementation and an implementation accelerated with PCIe-attached FPGAs, show that the network-attached FPGAs improved in latency and throughput. Additionally, network performance was compared with bare metal servers, virtual machines, and containers [76] with results showing orders of magnitude better for the FPGA prototype. To further improve the usability of the platform, continuous developments have been made to integrate MPI into the system [77], [78]. An in-depth study of FPGA cloud computing architectures is available in [79] and [80].

### D. GENERAL-PURPOSE CLUSTERS

Overspecialized systems tend to constrain the potential of reconfigurable hardware in favor of optimizing performance or costs. Nevertheless, general-purpose clusters are addressed by a larger group of projects seeking to change the programming paradigm. These clusters, rather than being a general purpose in the broad sense of the word, serve as experimental platforms to test solutions to all heterogeneous supercomputing challenges, ranging from network to user experience.

One of the first projects was the Reconfigurable Computing Cluster (RCC) [81] in the early 2000s. It was a multi-institution investigation project that explored the use of FPGAs to build cost-effective petascale computers, with its main contribution being the introduction of microbenchmarks for software, network performance, memory bandwidth, and power consumption. To evaluate each test *Spirit*, a cluster consisting of 64 FPGA nodes was built. Each node had a Virtex 4 FPGA with 2 Gigabit Ethernet ports, 8 DIMM slots for onboard RAM, and 8 MGTs for the board-to-board interconnection [82] using the Aurora protocol [83].

For internode communication, a configurable network layer core was developed as part of an Adaptable Computing Cluster project [84]. It consists of a network switch implemented in the FPGA acting as a concentrator for the router.

Considering that the head node is a workstation, a message-passing interface (MPI) approach offered the flexibility that the cross-development environment required. A custom compiler based on GNU GCC was built to support OpenMPI and its Modular Component Architecture (MCA) [85] which was adapted to support the high-speed network. A software infrastructure based on a Linux system allowed users to access, manage, and configure all nodes of the cluster via SSH [86].

Similar to the *RCC* project, the FPGA High-Performance Computing Alliance (FHPCA [87]) was established in 2005 with the *Maxwell* supercomputer [88]. The *Maxwell* CUs were built on a standard IBM BladeCenter chassis, in which an Intel Xeon and 2 FPGAs were interfaced via PCI-X. Additionally, an FPGA-dedicated network is available via MGTs without routing logic, given the nearest-neighbor scheme. By supporting standard parallel computing software, structures, and interfaces, it sought to disrupt the HPC space without causing significant friction.

To facilitate the development of applications targeting *Maxwell*, the Parallel Toolkit (PTK) [89] was developed. It included a set of practices and infrastructure to solve issues such as associating tasks with FPGA resources, segmenting the application into bitstreams, and managing code dependency. PTK provided a set of libraries where common standard interfaces, data structures, and components were defined.

Similarly, *Cube* was created to explore the scalability of a cost-effective massive FPGA experimentation cluster for real-world applications. It consisted of 8 boards that host a matrix of 8 by 8 Xilinx FPGAs [90] forming a cluster of 512 FPGAs, as shown in Figure 5. It features a single configuration of multiple data-programming paradigms that allowed all FPGAs to be configured with the same bitstream in a matter of seconds. The FPGAs were interconnected in a systolic array that reached up to 3.2 Tb/s inter-FPGA bandwidth offering significant advantages as it simplified the programming model and greatly relaxed the requirements of the PCB layout.

Simultaneously, *Quadro Plex (QP)* [91], a hybrid cluster was introduced. It was composed of 16 nodes, each consisting of one AMD CPU, 8 GB of RAM, 4 NVIDIA Quadro GPUs, and one Xilinx Virtex 4 Nallatech FPGA accelerator. The nodes were interconnected using Ethernet and Infiniband. Cluster communication was managed using OpenFabrics Enterprise Distribution software stack. The complete system occupied four 42U racks, consumed 18 kW, and had a theoretical performance of 23 TFLOPS. CUDA was used for GPU development, and the FPGA workflow completely relied on the Xilinx ISE design suite [92].

Several applications were developed, showing that there were substantial difficulties in taking advantage of an entire system. Applications would only use a combination of CPUs and GPUs or CPUs and FPGAs. A framework for easing the porting of applications and providing a compatibility layer for different accelerator workflows, called Phoenix [93] was developed.

In the same spirit, *Axel* [94] was built, consisting of 16 nodes. Each node had an AMD CPU, an NVIDIA Tesla GPU, and a Xilinx Virtex 5 FPGA occupying a 4U full-scale rack. All CEs were connected to a common PCIe bus for intranodal communication and between nodes in a Gigabit Ethernet network. Considering the high latency and nondeterministic nature of Ethernet, a parallel network using the 4 MGT of the FPGA was also available.

**TABLE 4.** Data center FPGA clusters' contributions reported power and performance improvement.

| Work | Contribution | Power | Performance |
|---|---|---|---|
| Catapult 2014 [22] | An FPGA-based cluster to accelerate the Bing web search engine ranking service. | Peak power consumption of 287 GFLOPS/W when running large deep neural network models with high device utilization | Twice the improvement in search throughput and a 29% reduction in the latency delay to process the search. |
| IBM Cloud FPGA 2015 [73] | Disaggregated FPGAs to increase computing density in racks for multi-tenant cloud systems | An order of magnitude more efficient than the MareNostrum 4 supercomputer | Peak performance of 344 millions of force interaction calculations per second for the N-body problem |

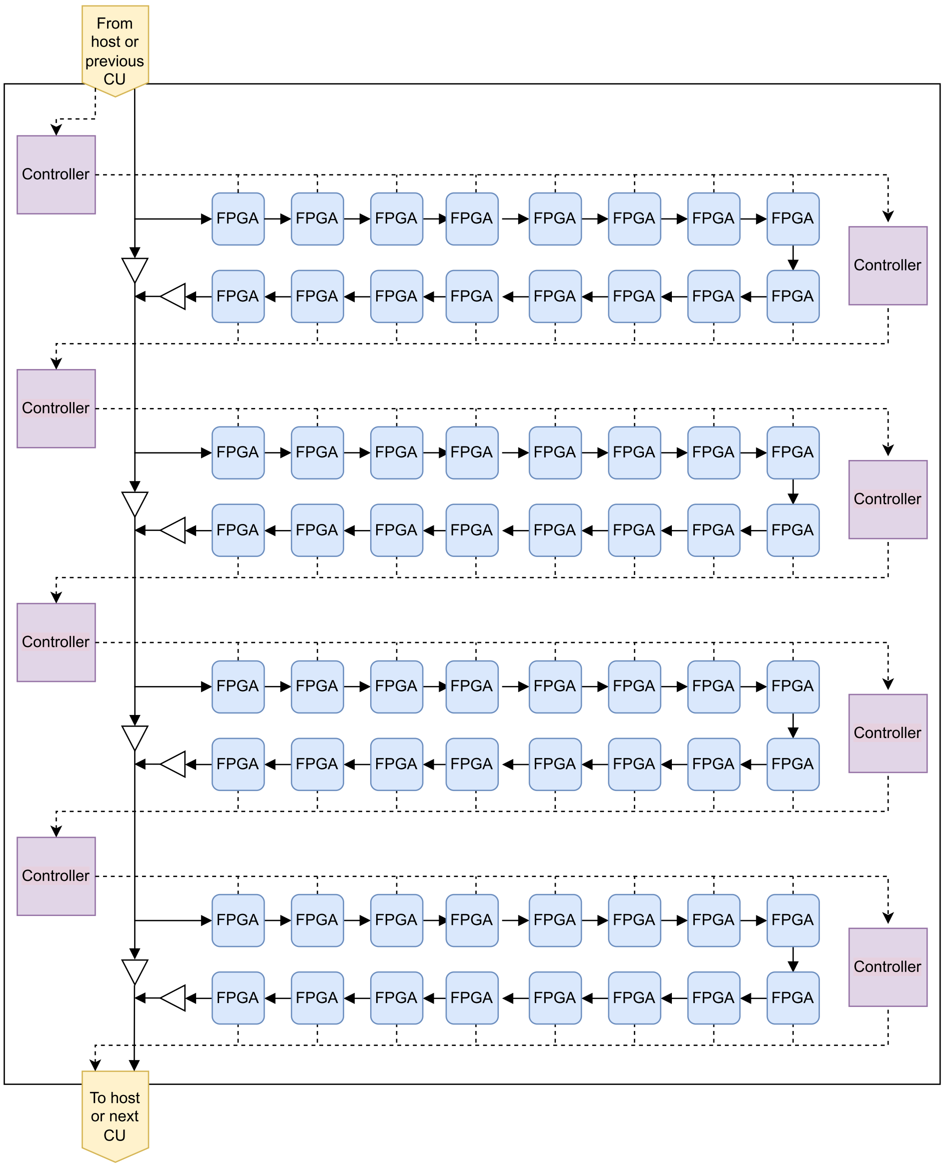


**FIGURE 5.** Cube [90] computational unit (CU) showing the configuration controllers in purple. Dotted lines show the control and configuration bus and solid lines show the data path.

The cluster was managed remotely from the central node using the Torque [95] resource manager and the Maui [96] scheduler. A custom resource manager (RM) was responsible for managing GPUs and FPGAs. For this to be feasible, all *Axel* programs needed to allocate part of the resources in the CEs to interface with the RM runtime API. Using an IPC message queue framework, CEs communicated their state to the head node. The central node collected information from all nodes with the help of the RM and prepared a script to submit the jobs to Torque. Communication between tasks in different nodes was performed via OpenMPI using Gigabit Ethernet.

To implement an application in *Axel*, users would provide a data flow graph and hardware abstraction model. A MapReduce framework then rewrites the application for partitioning the analysis into tasks. These tasks are assigned to the corresponding CEs based on the targeted attributes.

*Axel* also introduced an architecture classification for heterogeneous systems based on uniformity, shown in Figure 6. Following this classification, *Axel* is a Non-Uniform Node Uniform System (NNUS) architecture. This means that all nodes are equal but are built with different CEs. The advantage of this architecture is that the single-program multiple-data (SPMD) programming paradigm can be implemented easily. *Axel* also brought to light the need to reduce the design time and implementation time of FPGA, possibly by parallelizing the process to use heterogeneous clusters to optimize its own executable. Furthermore, it showed that design exploration tools were also lacking and essential for automating the performance estimation and code generation for multiple accelerators.

In 2010, *Novo-G* was presented as an experimental research cluster [97] consisting of 68 compute nodes built with COTS components. Its purpose was to help understand and advance the performance, productivity, and sustainability of future HPC systems and applications focusing on the sustainability problem of current HPC systems using three different PCIe Intel FPGA boards: 24 nodes with 192 Stratix III FPGAs boards, 12 nodes with 192 Stratix IV FPGA boards, and 32 nodes with 128 Stratix V.

*Novo-G* has been used for several acceleration projects, ranging from biology to finance. One aspect all applications have in common was being embarrassingly parallel and, therefore, naturally scalable. All of these applications were developed using the software offered as part of the *Novo-G* platform, and the results showed an enormous speed-up compared to CPU clusters.

*Chimera* was the first work to focus on implementing an algorithmic FPGA and GPU pipeline. The *Chimera* cluster [17] was built using commercial components to explore alternative solutions to the computational constraints found in astronomy and provide access to high-performance computing hardware for inexperienced users. The system is formed by CUs equipped with one CPU for management tasks, which is interfaced with 3 NVIDIA Tesla GPUs and 3 Altera Stratix IV FPGAs through PCIe via a backplane. Communication could always be considered a bottleneck, but in this case, it is clear that this limitation is directly related to the algorithms implemented in the entire system and the way each CEs interacts with the others.

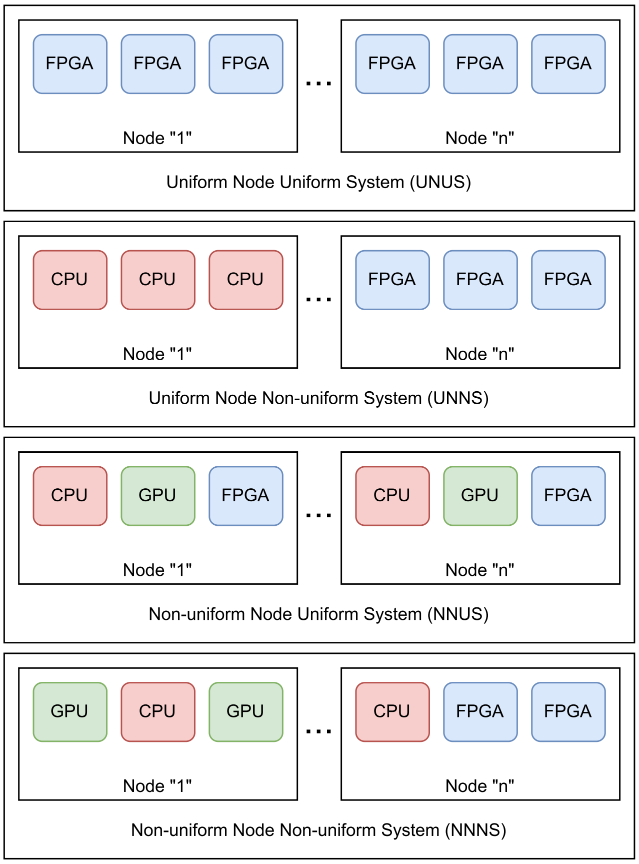

FPGA
FPGA
FPGA
Node "1"
...
FPGA
FPGA
FPGA
Node "n"
Uniform Node Uniform System (UNUS)
CPU
CPU
CPU
Node "1"
FPGA
FPGA
FPGA
Node "n"
Uniform Node Non-uniform System (UNNS)
CPU
GPU
FPGA
Node "1"
CPU
GPU
FPGA
Node "n"
Non-uniform Node Uniform System (NNUS)
GPU
CPU
GPU
Node "1"
CPU
FPGA
FPGA
Node "n"
Non-uniform Node Non-uniform System (NNNS)

**FIGURE 6. Axel [94] node or computational unit (CU) classification showing possible uniform and non-uniform node and system configurations for heterogeneous clusters of CPUs, GPUs, and FPGAs.**

The success of *Novo-G* and the advancement of technology have allowed *Novo-G* to be upgraded to *Novo-G#* [98]. The cluster is made up of Gidel ProceV accelerators that house Stratix V FPGAs, two 8 GB DDDR3, and 32 Mbits of SRAM memory. The boards were interconnected by grouping 24 transceivers into six groups to support a torus topology with a total bandwidth of 300 Gb/s. The physical connection is done with fiber optics using QSFP+ modules. The data are transmitted via packets through a configurable single-level router network. This allows one to instantiate as many routers as necessary to service the ports and increase the internal bandwidth at the expense of the hardware resources. The network flexibility enables users to experiment with a variety of routing modalities, depending on the requirements of the application. *Novo-G#* nodes support three communication blocks: a Low Latency, a Custom block, and Interlaken [99] to allow the optimization of the physical layer depending on the application.

A common problem in custom computing is the lack of software development tools to help users build applications. To solve this problem, the *Novo-G#* team developed a modified Altera OpenCL to provide extended support for the 3D torus network present in the cluster.

An important aspect that most clusters left out, besides those focused on communication, was the interface with the physical world. This is the empty space that the *Axiom* platform [100] seeks to fill with a custom scalable cluster based on a board with a Xilinx MPSoC (Multiprocessor System on Chip) supporting the Arduino interface.

The MPSoC has an FPGA fabric, four 64-bit ARM cores for general-purpose applications, and two 32-bit ARM cores for real-time applications in the same die. Four USB-C ports managed by the FPGA MGTs are available for interconnecting the boards. A custom network interface (NI) in the FPGA provides support for all communications, allowing users to focus on their applications written on an OpenMP extension called OmpSs. The NI is divided into six main groups: a data mover that deals with DMA transfers, RX and TX controllers, and FIFOs to cache packets. A router is interfaced with each NI and is responsible for handling the USB-C channels, monitoring the network, and establishing virtual circuits.

As part of the *Axiom* project, a custom software stack [101] consisting of multiple layers was also developed. Its foundation is a distributed shared-memory (DSM) architecture. The main advantage of this approach is that it allows applications to directly address physical memory by transparently relying on an OS network. Several tests [102] and benchmarks have validated the effectiveness of the platform, pushing the project forward into IoT and edge computing [103].

Progress in this field has led to the creation of the Xilinx Adaptive Compute Clusters (XACC) [104] group under the Xilinx Heterogeneous Accelerated Compute Clusters (HACC) [105] initiative. This industry and academic collaboration focuses on the development of new architectures, tools, and applications for next-generation computers.

As part of this initiative, several clusters were built at some of the world's most prestigious universities in Switzerland, the USA, Germany, and Singapore. At the Paderborn University's National High-Performance Computing Center (PC2), high-performance clusters *Noctua* [106] and *Noctua 2* [107] were built to provide hardware to accelerate research on computing systems with high energy efficiency.

The *Noctua 2* cluster was designed to fit common server racks and be compatible with the network industry standards. It has 36 nodes with 2 AMD Milan. A combination of 48 Xilinx Alveo and 32 Intel Stratix 10 GX FPGAs comprised the reconfigurable computing part of the cluster. Each Stratix node has 4 pluggable QSFP+ at 40 Gb/s and each Alveo has 2 QSPF+ at 100 Gb/s links and depends on Intel tools, such as oneAPI [108], OpenCL, and DSP Builder. A specific optical switch is used to build a configurable point-to-point network between all FPGAs.

More recently, *Enzian* [109] was developed as a scalable platform to fill the void left by industry-specific hybrid platforms. The reason behind Enzian provides a general, open, and affordable platform for research on hybrid CPU-FPGA computing, escaping the niche of specific-purpose hybrid platforms by providing a lot of flexibility. Explicit access to coherence messages, thermal and power monitoring, and an open baseboard management controller (BMC) allows for research that is not possible in any current commercial systems.

Likewise, *UNILOGIC* [110] presented a new approach, this time from the management of the cluster by introducing a Partitioned Global Address Spaces (PGAS) parallel model to heterogeneous computing. This allows hardware accelerators to directly access any memory location in the system, and locality makes coherency techniques unnecessary, greatly simplifying communication. By integrating Dynamic Partial Reconfiguration (DPR) into the framework, accelerators can be installed on the go. The *UNILOGIC* architecture was evaluated on a custom prototype consisting of 8 interconnected daughter boards, each with four Xilinx Zynq Ultrascale+ MPSoCs and 64 Gigabytes of DDR4 memory, yielding better energy and computing efficiency than conventional GPU or CPU parallel platforms.

In 2022, the supercomputer Cygnus [111] was updated [112] to follow a multi-hybrid accelerators approach based on GPUs and FPGAs. 32 Albireo nodes were added to Cygnus, each consisting of 4 NVIDIA V100 GPUs and two Intel Stratix 10 FPGAs. Similar to previous systems, a dedicated FPGA network was created with a 2D torus topology with improved stream capabilities, called CIRCUS [113]. Collaboration between the FPGAs and GPUs is achieved by using a DMA engine in the FPGA that accesses the GPU directly, bypassing the CPU, and offering almost double the throughput.

Finally, *Fugaku* [114], the first supercomputer to win all four categories in the Top500, presented a prototype FPGA cluster, *ESSPER* [115]. Motivated by the impressive continuous improvements in FPGAs regarding energy and performance, a cluster of 8 nodes, each with two Intel Stratix 10 FPGAs, was built and tested. This cluster was interfaced with Fugaku using a novel approach called loosely-coupled, where a host-FPGA bridging network provides interoperability and flexibility to all nodes in Fugaku.

### *E. COMMUNICATION SYSTEMS INFRASTRUCTURE*

Another field of application where clusters of FPGAs are relevant is the emulation of communication system infrastructure. The most important difference with manycore emulation is the need to interface with analog systems. This requirement implies providing additional external ports to interface with radio front-ends.

One of the first implementations was the *Berkeley Emulation Engine (BEE)* [117] in 2003. Its main purpose was to support design space exploration for real-time algorithms, focusing mainly on data-flow architectures for digital signal processing.

*BEE* was designed to emulate the digital part of telecommunication systems and to provide a flexible interface for radio front-ends. Computations are performed inside *BEE* Processing Units (BPU). Each BPU has a main processing board (MPB) and 8 riser I/O cards for 2400 external signals. The MPBs are the main computing boards hosting 20 Xilinx Virtex FPGAs, 16 zero-bus turnaround (ZBT) SRAMS, and 8 high-speed connectors. FPGAs on the periphery of the board have off-board connectors to link other MPBs. A hybrid network consisting of a combination of a mesh network and partial crossbar, called a hybrid-complete graph and a partial crossbar (HCGP) [118], was implemented. A single-board computer (SBC) running Apache web services over Linux allows users to deploy their applications and perform configuration and slow control tasks.

To take full advantage of the platform, an automated high-level workflow was used [119] that relied on MATLAB and Simulink to develop the main hardware blocks. The BEE compiler then processes the output and generates the required VHDL files for the simulation and configuration of the system. A time-division multiple access (TDMA) receiver was fully implemented to satisfy real-time requirements and validate the workflow.

Following the *BEE* success, the *BEE2* [120] was conceived as a universal, standard reconfigurable computing system consisting of 5 Virtex 2 FPGAs, each with 4 DIMM connectors for up to 4GB of RAM. Four FPGAs are available for computing, and one was reserved for control tasks. Pivoting away from the HCGP, an onboard mesh was implemented between the 4 computing FPGAs. Using high-speed links, it was possible to aggregate the 5 FPGAs and use them as a single, larger FPGA. The workflow remained almost the same for *BEE2*, with the main change being the use of a computational model of synchronous data flow for both the microprocessor and FPGA.

To overcome the shortcomings of *BEE2* and take advantage of the already validated Spirit architecture [121], a digital wireless channel emulator (DWCE) [122], [123] was developed. It consisted of 64 nodes in the same way as Spirit, but with valuable upgrades to demonstrate the capabilities of FPGA clusters with military radios. Its capabilities improved with an upgraded FPGA, additional 2 FMC connectors, and the adoption of a standard MicroTCA.4 form factor.

Considering the possible improvements to BEE and because it was being developed as part of the research accelerator for multiple processors (RAMP) community [124], a fast response was presented in the form of *BEE3* [116]. The development of *BEE3* differed from previous iterations and successfully demonstrated a new collaboration methodology between industry and academia [125].

The architecture of *BEE3* changed substantially from that of its predecessor by removing the control FPGA and introducing a control module on a smaller PCB. Another important aspect worth highlighting is that, for the first time, a PCB was intentionally developed to support different FPGA parts, all interconnected using a DDR2 interface in a ring topology.

The *BEE3* prototype had approximately 30 collaborators, most of whom were professionals with extensive knowledge of CAD. Relying on industry specialists for PCB design has resulted in simpler and more reliable PCBs within a shorter project time horizon. In addition, it was possible to parallelize the design process, allowing the academic community to focus on firmware development.

The BEE collaboration presented its final iteration in 2010, consisting of *BEE4* and *miniBEE* [126], [127]. BEE4 was updated to support Virtex 6 FPGAs and up to 128 GB of DDR3 RAM per module. The QSHs were removed in favor of FMC connectors to support a wider range of mezzanine boards. BEE4 was built around the Honeycomb architecture using the Sting I/O intermodule communication protocol. The design tools were further refined to include Nectar OS and BeeCube Platform Studio in MATLAB/Simulink, which are unfortunately proprietary. However, being a proprietary system did not discourage its use in academia [128]. The success of *BEECube* attracted further interest from the industry, and was bought by National Instruments in 2015 [129]. Today, it is a part of the FlexRIO [130] line-up, and software development is supported by NI tools. From this point onward, almost all implementations depend on commercially available emulation platforms.

To demonstrate the scalability of such implementations, the world's largest wireless network emulator was built, *Colosseum* [131], which can compute workloads of 820 Gb/s and perform 210 T operations per second. It was formed by CUs that consisted of three FPGAs in a chain. The outer FPGAs were used to interface with the radios and provide some processing. The central FPGA is dedicated to digital signal processing. Commercially available solutions were selected to avoid complications when designing the custom board. For the radio-attached FPGAs, 128 USRP-X312 [132] software-defined radios were used. Each provides the analog interfaces required for the antennas, along with a Kintex 7 FPGA. As dedicated processing FPGAs, 16 NI ATCA-3671 [133] modules were used, each hosting 4 FPGAs. The 64 processing FPGAs were interconnected in a $4 \times 4 \times 4$ HyperX topology [134] which allowed the data to be efficiently distributed for processing.

The NI modules are based on the BEE architecture and support the same development tools. Given the complexity of the system, a Python data-flow emulator [135] was built to confirm the topology and architecture of the system. It is possible to confirm the latency of the system by providing models of the implemented components and topology.

Another notable contribution of this study is the proposal of a data flow methodology [136]. It comprises three guiding principles that highlight the issues present in other implementations. The first principle is the use of a unified interface for modular components to favor portability. Second, when dealing with heterogeneous systems, the suggested approach is asynchronous processing to decouple operations from time and favor parallelization. Finally, based on design best practices, solutions are urged to be vendor-independent.

**TABLE 5.** **General-purpose clusters' contributions, reported power and performance gains.**

| Work | Contribution | Power | Performance |
|---|---|---|---|
| Maxwell 2007 [88] | One of the first general-purpose FPGA supercomputers. | | 10 to 100 times faster than Intel Xeon CPUs. |
| Spirit 2007 [81] | Prototype FPGA cluster to explore methodologies for future PetaFLOP computers. | | |
| BEE3 2009 [116] | Industry and academia's reconfigurable supercomputing ecosystems. | | More than 1 order of magnitude faster when simulating systems with 16 processors compared to a software implementation. |
| Cube 2009 [90] | Low-cost architecture for scalable FPGA clusters with a single-configuration multiple-data programming paradigm for systolic arrays. | 600 times more efficient than a 180 dual Xeon quad-core cluster. | 8 times faster than a 359 high-end CPU cluster. |
| QP 2009 [91] | A heterogeneous cluster to explore multi-accelerator collaboration within a new framework. | | FPGA part speed-up of 48.4 times with respect to a 16 AMD dual-core cluster. |
| Axel 2010 [94] | MapReduce framework for asymmetric parallel computing on heterogeneous clusters. | A 19 times increase in efficiency was obtained for the FPGA and GPU implementation when compared to CPU | The FPGA and GPU collaborative approach resulted in a 44.5 times speedup for GARCH asset simulation compared to CPU-only implementation. |
| Novo-G 2010 [97] | Experimental cluster to investigate productivity, performance, and sustainable architectures. | 8kW | More than 800 times faster on bioinformatics applications over an AMD Opteron CPU. |
| Chimera 2012 [17] | Evaluation of a scalable heterogeneous desktop platform in the context of Berkeley dwarfs. | | Peak performance 4.3 Gflop/J for the GPU part and 25 Mflop/J for the FPGA part. |
| Novo-G# 2016 [98] | Reconfigurable cluster with reprogrammable interconnects to explore ways to minimize communication latency between nodes. | | For 3D FFT, the cluster matches the latency of 512-core BlueGene/Q. |
| Axiom 2017 [100] | Scalable reconfigurable physical computing for low-cost applications. | 8 times more efficient for iris plus voice recognition than software implementation. | 1.5 times faster than a pure software implementation. |
| Noctua 2 2019 [106] | Data-center-grade FPGA cluster with optically switched interconnects for dynamic topology configurations. | | Peak performance of 8.2 Petaflops. |
| Enzian 2020 [109] | Open hardware platform for CPU / FPGA systems with coherent access to asymmetric cache and fine-grained profiling. | | |
| UNILOGIC 2020 [110] | First implementation of a Partitioned Global Address Spaces parallel computing model for heterogeneous clusters. | At least 10 times more efficient than CPUs or GPUs. | More than 2.5 times faster than CPUs or GPUs. |
| Cygnus Albeiro 2022 [112] | First hybrid cluster with direct collaboration of FPGA-GPU over DMA. | | Radiative transfer simulation results show a speedup of 12.8x for the FPGA-GPU collaborative approach over the GPU-only solution. Peak performance 2.4 PFLOPS. |
| ESSPER 2023 [115] | Loosely-coupled FPGA cluster prototype upgrade of the second largest supercomputer in the world. | | |

**TABLE 6. Communication systems emulation clusters' contributions, reported power and performance gains.**

| Work | Contribution | Power | Performance |
|---|---|---|---|
| BEE 2003 [117] | Custom multi-chip FPGA emulation engine for ultra-wideband and multi-channel multi-antenna radio research. | | More than 100000 times faster than the best simulation software at that time. |
| BEE2 2005 [120] | High-level block diagram design environment based on MathWorks Simulink and Xilinx System Generator library for one unified computation model for microprocessors and FPGAs. | Two orders of magnitude more efficient than Intel Pentium 4 CPUs. | 10 times more computing throughput than a DSP-based system. |
| DWCE 2011 [122] | FPGA cluster for decentralized signal processing for scalability of radio emulation. | | Same performance as 15237 CPUs cores running at 2 GHz |
| Colosseum 2021 [131] | A scalable architecture for implementing real-time channel emulators using software-defined radios and FPGA with a hardware-in-the-loop approach. | | Emulation of up to 65535 wireless channels with an area of up to 1 km$^2$. |

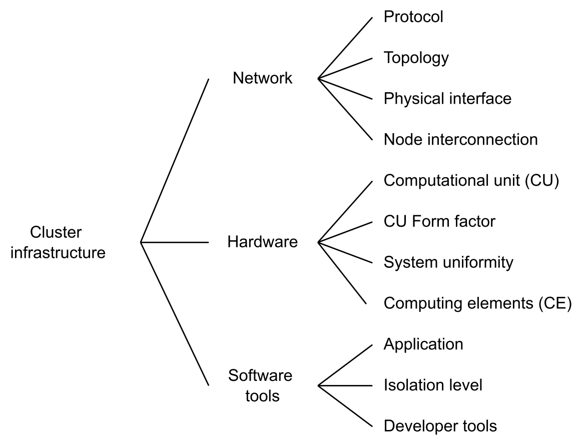


**FIGURE 7. Main concepts for the proposed classification of clusters.**

## II. CLASSIFICATION

After studying each of the works described above, it was possible to identify common elements. These elements reflect the decisions made by the designers when conceiving each cluster. Given that heterogeneous computing is broad and complex, until now no universal methodology has been developed to design a cluster. A classification system was proposed in [94] based on the uniformity of the system and its nodes.

We proposed segmenting the cluster infrastructure into three main components, as shown in Figure 7. The first aspect is the network. This covers the physical interfaces chosen to connect the nodes, logic protocols, and topologies. Another important aspect to consider is the hardware available in the CU. Each CU can have more than one CE type. Finally, software tools that allow the cluster to be securely available to users for development were considered. They encompass isolation tools that protect hardware from misbehavior, which are discussed in [137], [138] but go beyond programming languages, APIs, libraries, etc. All the cited works provide an overview of the tools available to the user and the intended workflow. Depending on the target application these tools vary in scope, flexibility, and complexity. With a study of all previous contributions, it is possible to build a wide base that helps understand the greatest challenges, future trends, and real capabilities of heterogeneous supercomputing.

### A. NETWORK

A cluster is no more than a set of computational elements (CE) that collaborate toward a common goal. The collaboration method and its means are crucial for ensuring implementation efficiency. The means of collaboration branch out from the hardware interfaces to the communication protocols and, ultimately, the schedulers or other methods of synchronization. With this consideration, we can draw a line between systems that delegate communication tasks to an external entity and those that incorporate the stack. Another important aspect of the interconnection is how it is handled. In high-speed stream computing, it is desired that the communication be established as direct data channels with back-pressure, and this particular aspect is difficult to replicate with purely routed networks.

Table 7 shows several aspects that distinguish the implementations concerning network infrastructure for all the works presented in Section I. The manner in which nodes are connected is discriminated according to the existence of any additional hardware that processes, redirects, or interprets a stream of data or packages between adjacent nodes as an indirect interconnection. This implies that a direct interconnection is such that one node can interact directly with the

**TABLE 7. Network infrastructure.**

| Work Title | Interconnection | Topology | Interface | Protocol |
|---|---|---|---|---|
| RAW 1997 [32] | Direct | TLM | GPIO | Custom |
| BEE 2003 [117] | Indirect | TLM | GPIO | Custom |
| BEE2 2005 [120] | Direct | Mesh | MGT | Internal DDR, External Infiniband |
| COPACOBANA 2006 [53] | Direct | Star | GPIO | Custom |
| FAST 2006 [33] | Direct | Star | GPIO | Custom |
| Janus 2006 [51] | Indirect | 2D Torus | GPIO | Internal Custom, External GbE |
| Maxwell 2007 [88] | Direct | 2D Torus | MGT | Custom |
| Spirit 2007 [81] | Direct | 3D Torus | MGT | Aurora |
| BEE3 2009 [116] | Direct | Ring | GPIO, MGT | Internal DDR, External GbE |
| Cube 2009 [90] | Direct | Systolic | GPIO | Custom |
| QP 2009 [91] | Indirect | Star | MGT | Ethernet, Infiniband |
| Axel 2010 [94] | Indirect | Star | MGT | Ethernet, Infiniband |
| BEE4 2010 [127] | Direct | Ring | GPIO, MGT | Internal DDR, External GbE |
| Formic 2010 [42] | Direct | 3D Torus | MGT | Sata |
| Novo-G 2010 [139] | Direct | Systolic | MGT | Custom |
| RAPTOR 2010 [35] | Indirect | Star | MGT | PCIe |
| Tse, et al. 2010 [57] | Indirect | Star | MGT | Ethernet |
| DWCE 2011 [123] | Direct | 3D Torus | MGT | Aurora |
| Bluehive 2012 [59] | Direct | Mesh | MGT | PCIe |
| Chimera 2012 [17] | Indirect | Mesh | MGT | PCIe |
| Cuteforce 2013 [56] | Indirect | Star | MGT | Infiniband |
| Catapult 2014 [22] | Direct | 2D Torus | MGT | Serial Lite III |
| Janus II 2014 [63] | Indirect | 2D Torus | MGT | Internal Custom, External GbE |
| Superdragon 2015 [58] | Indirect | Star | MGT | Infiniband |
| Novo-G# 2016 [98] | Direct | 3D Torus | MGT | Custom |
| IBM cloud FPGA 2017 [75] | Indirect | Star | MGT | GbE |
| Axiom 2017 [100] | Direct | Mesh | MGT | Custom |
| ARUZ 2018 [25] | Indirect | 3D Torus | GPIO, MGT | Internal Custom, External GbE |
| Green Flash 2018 [140] | Indirect | TLM | MGT | Internal PCIe, External GbE |
| Noctua 2 2019 [106] | Direct | Flexible | MGT | Custom |
| Astrobyte 2020 [66] | Direct | Mesh | MGT | Custom |
| Enzian 2020 [109] | Direct | Flexible | MGT | Internal PCIe, External GbE, Custom |
| UNILOGIC 2020 [110] | Direct | Hypercube | MGT | AXI Chip-2-Chip |
| BiCoSS 2021 [67] | Direct | BFT | GPIO | Custom |
| Colosseum 2021 [131] | Direct | Ring | GPIO, MGT | Internal DDR, External GbE |
| Cygnus Albeiro 2022 [112] | Direct | 2D Torus | MGT | CIRCUS |
| ESSPER 2023 [115] | Direct | 2D Torus | MGT | Internal Avalon-ST, External custom |

nearest neighbor without the need for additional networking hardware, excluding physical interfaces. In these implementations, network services are provided by in-fabric routers and switches, which allow users to experiment with different protocols at the expense of resources. This is particularly crucial in implementations targeting heavy communication problems that require low latency. By no means, the dependence on external hardware imposes a disadvantage because recent implementations show that it is capable of extending scaling capabilities without affecting performance, as in the case of [25], which effectively interconnects thousands of CEs. As shown in [141], adding dedicated network hardware increases the latency by a constant factor. To determine the impact on performance, a ring topology was implemented using the E40G protocol. The experiments showed that the performance depended on the size of the packet. For smaller packets, the latency was dominant, tipping the scale in favor of direct interconnection, but for larger packets ($> 1$ MB), the switched implementation offered an improvement of approximately 5%.

To compare the impact of both network connections, a leaf-spine topology was implemented for the switched network, and a ring topology for direct interconnection. The switched network was modeled for 2048 FPGAs using 64 radix switches. The ring topology was simulated for a direct network. These simulations showed that for a small message size ($\approx < 1$ MB), a direct network offers a shorter transmission time than a switched network, regardless of the number of nodes. In contrast, larger payloads ($> 227$ MB) benefit from a switched network, but only up to 1024 nodes when the direct network transmission time catches up. These results show that one approach is not necessarily better, but that it comes down to the specificity of each cluster.

Similarly, the topology of the clusters is a decisive design factor. Given that nodes are desired to have the highest throughput with the lowest latency, most researchers have opted for a tightly interconnected topology such as a mesh or two-level mesh (TLM). These are great at providing a consistent distance between the nodes in the system, but they strongly affect larger systems. Another popular topology is torus, 2D, or 3D. It has the advantage of limiting the longest distance between nodes; however, as mentioned before, this distance continues to grow as more nodes are added to the system. The system should also keep a uniform shape and nodes should be added to fill columns or rows to avoid introducing inconsistencies to latency. Later works, such as Noctua and Enzian, are not bound by a fixed topology. In particular, Noctua's infrastructure provides an optical switch capable of implementing different topologies in runtime based on user requests. Naturally, given that this is an external device, it can be implemented in any indirectly connected cluster. For some of the directly interconnected clusters, it may be possible to add an external switch, but only if a standard protocol and interface are used, such as Novo-G and Novo-G#.

The strong relationship between the interface and the protocol is hard to break, and it is rare to find a reason good enough to do this. For the same reason, most MGT implementations are based on the Aurora protocol or similar for the physical layer, and the data link relies on Ethernet. However, Bluehive challenges this reasoning by implementing eSATA over PCIe connectors because PCIe was the only high-speed connector available on the FPGAs nodes. Other particular design choices include the implementation of DDR over GPIOs in same-board communications; this is the case for BEE, BEE2, and BEE3. It can be appreciated that, considering the complexity of bringing up one of these systems, standard protocols, and interfaces have been more favored. This stems from the fact that proven technologies shorten design times, allowing developers to focus on other issue.

### B. HARDWARE

When studying a cluster's hardware, it is helpful to divide it into its computational units (CU). A CU is any entity that is available for computing and is the smallest independent functional part of the cluster. According to this definition, devices that act as pure network appliances, routers, or switches are not considered. A CU can be composed of multiple CEs. Over time, smaller CUs are preferred when dealing with general-purpose clusters, whereas specific problems can benefit from larger CUs with an ad hoc network topology.

In the context of *Axel*, [94], a classification structure was proposed. It focuses on identifying the nodes, which in this paper we refer to as CUs, to avoid confusion with network nodes, by their CEs, and the way in which they are distributed in the system. Four different types were considered, as shown

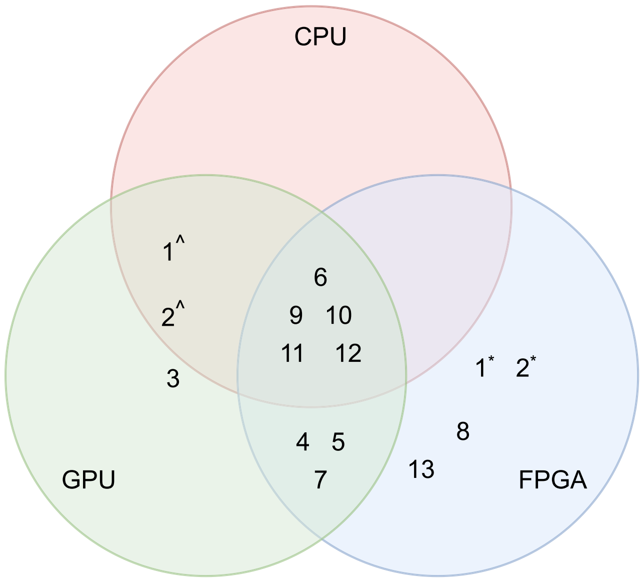


**FIGURE 8. 13 dwarfs (table 1) mapped-out according to highest affinity computational element (CE); the superscript ^ refers to floating point and ∗ to fixed point [17].**

in Figure 6. The Uniform-Node Uniform-System (UNUS) corresponds to a homogeneous cluster and is typically formed by CPUs. However, we can also find FPGA implementations such as Formic [39], Janus I [51], and Janus II [63]. This approach has the advantage that a single programming model can be applied to the entire system but is not restricted by it. Given that an FPGA is by nature a heterogeneous device, this is not the case. Uniformity significantly simplifies the management and maintenance of clusters. Interestingly, the advantages have not critically outweighed the disadvantages, given that several other studies have explored other more complex approaches. Performance-wise, there is a lot to win when dealing with non-uniform nodes or systems. As shown in Table 1, all the computing problems can be classified into 13 categories. By carefully studying the affinity between each category of problems and the resources encapsulated in each CE, ideal candidates to solve each problem can be found. This means that by mixing and matching, a heterogeneous cluster may offer performance advantages over a homogeneous cluster.

Figure 8 shows how each of the problems is mapped to CEs depending on their characteristics [17]. This map shows that there are clear advantages in using one type of CE over another, mainly between GPU and FPGA. This is further supported by the implementations of *Chimera* [17] and *Green Flash* [140]. In both cases, GPUs and FPGAs are intended to be used as collaborative CEs. However, this new paradigm makes development much more complicated, not because of the lack of tools for FPGA and GPU co-processing, but also because of the required radical change of mindset when leaving the traditional CPU plus accelerator context. This is reflected in the reported applications of *QP* [91] in which users used only a combination of CPU plus GPU or FPGA.

Table 8 shows the most relevant studies classified according to the characteristics of their CU. The *Axel* classification system was used to identify the uniformity of the node (CU) and system. The total number of CUs is also presented. Some studies have presented the architecture of a single node as a building block for a future cluster. These are considered relevant for their contribution to the study of heterogeneous workloads. The form factor of each work is shown in the respective column. Given that these are heterogeneous systems, different types of CE may be present at CUs, and sometimes even among CUs. Finally, Table 8 shows the total number of CEs implemented in the system.

As previously described, classification based on node (CU) and system uniformity is useful for understanding the programming paradigm. According to the previous definition of nodes, multiple CEs can be hosted on a single node. A balance between a crowded or simple node resides in the diversity of the CEs and network infrastructure. Diverse CEs in a single node allow for the highest resource availability per CU for developers, but it remains a challenge to interface all devices considering all the different ports. This leads to different form factors that directly impact the way the cluster scales, and more importantly, the availability of physical structures to hold the nodes in place and provide efficient cooling. Custom CU form factors usually host several CEs and can compromise not only scalability, but also fault tolerance. This is the case for *ARUZ* [25], where a single node hosts up to 11 FPGAs. In an unfortunate case where one or more CEs break down, the OS must be notified to circumvent these nodes or completely ignore them until they are fixed. In this regard, COTS clusters have a great advantage: the up-bring cost is mostly absorbed by the industry by providing tested and validated nodes for quick installation, which is the case for *Noctua 2* [107] and *Catapult* [22], among others. Some rare cases of industry and academia collaboration greatly benefit from COTS advantages with specific research-motivated modifications, as in *Novo-G*, [97] and *BEE3* [125].

## C. SOFTWARE TOOLS

Finally, each work discussed would be incomplete if there were no tools available to help users develop their applications. These tools provide different layers of isolation, ranging from templates that encapsulate internode communication to complete operating systems that manage multiple-user access. Each of the tools offers a degree of abstraction encapsulating all underlying details to offer services to the user or to a higher layer. The depth of the layer stack depends on several factors:

- Purpose of the cluster
- Degree of freedom intended for the user (isolation)
- Cluster flexibility

A stack of tools can be structured according to the services provided and required, as shown in Figure 9. First, we have the interface with the external worldatn a physical level. Naturally, we rely on electric signals controlled by internal gates, GPIOs, or MGTs. Typically, in HPC, this is not out for discussion, given that CPUs and GPUs have fixed interfaces, but FPGAs are not bounded by this. Thus, the communication layer can be either available for users to freely customize and test or fixed by the developers and provided as a service. In addition, we have actual CEs; these can be CPU, GPU, or FPGA custom cores. It is in this part where actual computing is performed, and users may be able to define the entities, or developers may provide programmable blocks. To interact safely with these block drivers, a file system and a scheduler may be provided as an operating system. This creates a safe space for users to build applications based on the hardware and communication services. At this level, users must rely on a programming language that describes how the underlying parts cooperate for the intended computation. Some studies have presented new programming languages that aim to capture the different programming paradigms in heterogeneous clusters. Tools that take the abstract description of the computation and transform it into instructions may be provided as a contained solution or along libraries and APIs to facilitate development. Another level of abstraction may be introduced, in which users interact with prebuilt blocks in a fixed context inside a GUI.

Table 9 shows a series of works with the development tools provided and the intended application. Depending on the scope of the application, user needs vary and may require deeper access to the system or more abstract tools. Most general-purpose clusters are intended to be used as research platforms. This requirement relaxes many management applications and abstraction layers that, in turn, must be provided to the user in other cases. As research could take part in the lowest level of communication, users may need the freedom to change the electrical standard of the GPIOs or the encoding of the MGT. These properties are only available if the user sees the platform as a bare metal solution, or if the development environment has a standardized way of defining communication devices. In any case, most systems avoid this by providing the user with a template that abstracts the communication layer. Specific application clusters seek to encapsulate most of the details such that the user faces only the challenges related to the application.

The flexibility of the software stack also depends on the platform's openness. In this regard, FPGA development frameworks have been significantly delayed as opposed to CPU and GPU. Currently, one can use complete open-source frameworks to develop applications for CPUs and GPUs, but FPGAs are radically different. One reason for this is that the stack of tools is fundamentally different. Instead of targeting fixed hardware through a well-known and well-defined instruction set architecture (ISA), FPGA tools target configuration memory with architecture-specific information. These architectural details tend to be industry secrets that force developers to rely on vendor tools with all their benefits and limitations. One of the most important limitations is the proprietary nature of some vendor tools. Efforts have been made to create completely open-source workflows

**TABLE 8.** Hardware architecture of computation units (CU) with respect to their computational elements (CE).

| Work Title | Classification | CU | Form factor | CE/CU | Resources |
|---|---|---|---|---|---|
| RAW 1997 [32] | UNUS | 5 | Custom | 64 | 320 Xilinx 4013 FPGAs, total 180 M CLBs, 735 kB RAM |
| BEE 2003 [117] | UNUS | 4 | Custom | 5 | 20 Xilinx XC2000E, total 20 k CLBs |
| BEE2 2005 [120] | UNUS | 40 | Custom | 5 | 200 Xilinx Virtex-2 Pro FPGAs, total 6.6 M CLBs, 200 PowerPC 504, 64.6 k DSPs; 800 GB DDR2 RAM |
| COPACOBANA 2006 [53] | UNUS | 20 | Custom | 6 | 120 Xilinx Spartan-3 S1000 FPGAs, total 490 k CLBs, 8.3 MB RAM, 2880 multipliers |
| FAST 2006 [33] | NNUS | 2 | Custom | 8 | 16 Xilinx Virtex-5 FPGAs, total 668.8 k CLBs, 12.8 MB RAM, 3 k DSPs; 627 MB SRAM |
| Janus 2006 [51] | UNUS | 16 | 2U half rack | 16 | 256 Xilinx Virtex-4 LX200 FPGAs, total 5 M CLBs, 238 MB RAM, 24.5 k DSPs |
| Maxwell 2007 [88] | NNUS | 32 | PCI-X | 3 | 32 Intel Xeon; 32 Nallatech HR101s with Xilinx Virtex-4 LX160, total 540.7 k CLBs, 3 k DSPs, 16 GB SRAM, 20 MB RAM; 32 Alpha Data ADM-XRC-4FXs with Xilinx Virtex-4 FX100, total 337 k CLBs, 64 PowerPC, 5.1 k DSPs, 29.7 MB RAM; 48 GB RAM |
| Spirit 2007 [81] | UNUS | 64 | Custom | 1 | 64 Virtex-4 FX60 total, 455 M CLBs, 33.4 MB RAM, 128 PowerPC 405, 8.1 k DSPs |
| BEE3 2009 [116] | UNUS | 64 | 2U full rack | 4 | 256 Xilinx Virtex-5 LX155T FPGAs, total 3 M CLBs, 243 MB RAM, 32 k DSPs; 64 GB DRAM |
| Cube 2009 [90] | UNUS | 8 | Custom | 64 | 512 Xilinx Spartan-3 S4000, total 2.2 M CLBs, 40 MB RAM, 49 k multipliers |
| QP 2009 [91] | NNUS | 16 | Desktop PC | 6 | 16 AMD dual-Core CPUs; 64 NVIDIA Quadro FX 5600 GPUs; 16 Nallatech H101 with Xilinx Virtex-4 LX160, total 8 GB SRAM, 270 k CLBs, 1.5 k DSPs, 10 MB RAM |
| Axel 2010 [94] | NNUS | 16 | 4U full rack | 3 | 16 AMD quad-core CPUs; 16 NVIDIA Tesla C1060 GPUs; 64 GB DDR2 RAM; 64 GB DDR3 RAM; 16 Xilinx Virtex-5 LX330T, total 414 M CLBs, 30 MB RAM, 3 k DSPs |
| BEE4 2010 [127] | UNUS | 1 | 2U full rack | 4 | 4 Xilinx Virtex-6 475T, total 148.8 k CLBs, 22.9 MB RAM, 8064 DSPs |
| Formic 2010 [42] | UNUS | 64 | Custom | 1 | 64 Xilinx Spartan-6 LX150T, total 9.5 M CLBs, 38 MB RAM; 216 MB SRAM; 8 GB RAM, 11.5 k DSPs |
| Novo-G 2010 [97] | NNNS | 68 | 4U full rack | 4, 8, 16 | 512 FPGAs (192 Strtix III E260, 192 Stratix IV E530, 128 Stratix V GSD8), total 239 M logic elements, 1.8 GB RAM, 18 k DSPs, 699 k multipliers |
| RAPTOR 2010 [35] | NNUS | 16 | Custom | 4 | 64 Xilinx Virtex-5 FX100T FPGAs, total 1 M CLBs, 93 MB RAM, 16.4 k DSPs, 128 PowerPC; 256 GB RAM |
| Tse, et al. 2010 [57] | NNUS | 8 | 2U full rack | 4 | 16 AMD Phenom 9650 quad-core; 8 NVIDIA Tesla C1060; 8 Xilinx Virtex 5 LX330T, total 103 M CLBs, 15 MB RAM, 3 k DSPs |
| DWCE 2011 [123] | UNUS | 64 | MicroTCA.4 | 1 | 64 Xilinx Virtex-6 475SX FPGAs, total 2 M CLBs, 5.7 MB RAM, 129 k DSPs |
| Bluehive 2012 [59] | UNUS | 4 | 8U full rack | 16 | 64 DE4 boards with Altera Stratix IV 230 FPGA, total 14.5 M logic elements, 137 MB RAM, 82.4 k multipliers |
| Chimera 2012 [17] | NNUS | 1 | Custom | 7 | 1 Intel i7 hexa-core; 3 NVIDIA Tesla C2070 GPUS; 3 Intel Stratix IV 530, total 16 M logic elements, 10 MB RAM, 3 k multipliers |
| Cuteforce 2013 [56] | UNNS | 15 | 2U full rack | 1 | 1 NVIDIA GeForce GTX 680 GPU; 14 Xilinx ML605 FPGA development boards, total 263 M CLBs, 26 MB RAM, 10.8 k DSPs |
| Catapult 2014 [22] | NNUS | 1632 | 1U half rack | 3 | 3264 Intel 12-core CPUs; 104 TB RAM; 1632 Stratix V D5 FPGAs, total 745 G logic elements, 13 TB DDR3 RAM, 5.2 M DSPs |
| Janus II 2014 [63] | UNUS | 16 | 2U half rack | 16 | 256 Xilinx Virtex-7 VX485T FPGAs, total 9 M CLBs, 1.5 GB RAM, 716.8 k DSPs; 4 TB DDR3 RAM |

**TABLE 8.** ***(Continued.)*** **Hardware architecture of computation units (CU) with respect to their computational elements (CE).**

| | | | | | |
|---|---|---|---|---|---|
| Superdragon 2015 [58] | NNUS | 64 | 2U full rack | 2 | 64 NVIDIA Testala C2050 GPUs; 64 Xilinx Virtex 5 LX330 FPGAs, total 1.7 M CLBs, 110 MB RAM, 12.3 k DSPs; 128 Intel Westmere hexa-core CPUs; 1.9 TB DDR3 RAM |
| Novo-G# 2016 [98] | NNUS | 64 | 4U full rack | 3 | 128 Xeon E5 CPUs; 512 GB DDR3 RAM; 64 Intel Stratix V FPGAs, total 44 M logic elements, 464 MB RAM, 251 k multipliers |
| Axiom 2017 [75] | NNNS | 16 | Custom | 1 | 16 Xilinx Ultrascale+ ZCU9EG, total 16 ARM quad-core A9 CPUs, 9.6 M system logic cells, 92 MB RAM, 40.3 k DSPs |
| IBM cloud FPGA 2017 [75] | UNUS | 16 | 2U full rack | 64 | 1024 Xilinx Kintex UltraSCALE KU060 FPGAs, total 743 G system logic cells, 6 GB RAM, 2.8 M DSPs; 16 TB DDR4 RAM |
| ARUZ 2018 [25] | UNUS | 2901 | Custom | 11 | 23040 Xilinx Artix-7 A200 FPGAs, 2922 Xilinx Zynq Z015, total 2922 ARM A9 dual-core, 421 M CLBS, 47.4 GB RAM, 17.5 M DSPs; 2.9 TB DDR2 RAM |
| Green Flash 2018 [140] | NNUS | 5 | ATX mini | 3 | 5 Xeon Phi CPUs 7210; 5 Intel Arria 10 SoC, total 3.3 M logic elements, 10 ARM dual-core A9, 25MB RAM; 5 Intel Arria 10 FPGAs total, 5 M logic elements, 33 MB RAM, 15 k multipliers |
| Noctua 2 2019 [106] | NNNS | 36 | 2U full rack | 5 | 72 AMD 64-cores Milan; RAM 18.4 TB; 48 Xilinx Alveo U280 FPGA, total 6 M CLBs, 384 GB HBM2, 433 k DSPs, and 1.5 TB DDR memory; 32 Stratix 10 total, 1 TB DDR RAM, logic elements 88 M, 976 MB RAM, 368 k multipliers |
| Astrobyte 2020 [66] | UNUS | 4 | Custom | 1 | 4 Intel Stratix IV GX FPGAs, total 2.1 M logic elements, 13.7 MB RAM; 2.5 GB SRAM, 4 k multipliers |
| Enzian 2020 [109] | NNUS | 1 | 2U full rack | 2 | Cavium ThunderX CN88XX; Xilinx Ultrascale+ VU9, total 2.6 M system logic cells, 47.8 MB RAM, 6.5 k DSPs |
| UNILOGIC 2020 [110] | UNUS | 16 | 1U full rack | 4 | 1TB DDR4 RAM; 64 Xilinx Ultrascale+ ZU9EG MPSoCs, total 38.4 M system logic cells, 320 MB RAM, 151 k DSPs |
| BiCoSS 2021 [67] | UNUS | 5 | Custom | 7 | 35 Intel Cyclone IV FPGAs, total 3.9 M logic elements, 17 MB RAM, 1.9 k multipliers |
| Colosseum 2021 [131] | UNUS | 16 | 2u rack | 28 | 128 Intel 48-core Xeon; 128 NVIDIA Tesla K40m; 128 Ettus USRP X310 with Xilinx Kintex-7 410T, total 4 M CLBs, 548 MB RAM, 197 k DSPs; 64 Xilinx Virtex-7 690T, total 3.4 M CLBs, 423 MB RAM, 1 TB DDR RAM, 230.4 k DSPs |
| Cygnus Albeiro 2022 [112] | NNUS | 32 | 4U full rack | 8 | 64 Intel Xeon; 128 NVIDIA Tesla V100; 64 Intel Stratix 10 GX2800 FPGAs, total 179.2 M logic elements, 128 GB DDR4 RAM, 737.3 k multipliers |
| ESSPER 2023 [115] | UNNS | 8 | 1U full rack | 4 | 16 Intel Xeon quad-core; 16 Intel Stratix 10 SX2800 FPGAs, total 44.8 M logic elements, 32 GB DDR4 RAM, 184 k multipliers, 16 ARM A53 quad-core |

such as F4PGA [8], in which experienced users can actively collaborate to improve the platform. Finally, an important aspect directly tied to the application is the level of flexibility provided by the cluster. Some applications can implement external hardware for optional data streams. This is the case for all the BEE implementations. Other domains of flexibility include the network topology and communication protocol of the cluster. The portability of the framework was described by the flexibility of the CEs. This means that the cluster CEs could potentially be updated or changed without requiring important modifications of the development tools, future-proofing them, and providing customization depending on the user's needs.

## III. OPEN PROBLEMS AND TRENDS

Supercomputing is a complex and fast-evolving field in which CPUs and GPUs have traditionally dominated the market. Several successful attempts have been made to introduce FPGAs in this context, such as F1 instances in Amazon and the IBM cloud FPGA service. The flexibility and energy efficiency of FPGAs strongly challenge CPU and GPU for the same computing tasks, further motivating research in this area.

The opportunities that FPGAs offer to heterogeneous computing are huge. As already shown in several studies [14], [142], [143], [144], [145], FPGAs can surpass CPU energy efficiency by orders of magnitude by relying on

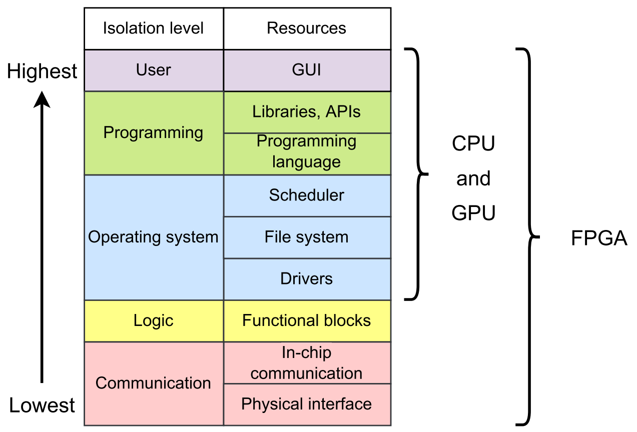


**FIGURE 9.** Tool stack divided into different levels depending on the user isolation from the hardware.

hardware-level customization. FPGAs also offer the highest degree of control to developers, allowing optimization at a logic level that is impossible with CPUs and GPUs. In addition, modern SoC-FPGAs offer internal high-speed connections that allow CPUs, GPUs, and FPGAs to interact on the same die, thereby reducing communication latency and power consumption.

However, as has been shown in this paper, having hardware does not mean that the problems are solved. One of the biggest obstacles to mass adoption is the lack of hardware abstraction and efficient synthesis tools, which increases development time [14], [144] compared to CPUs and GPUs. To identify the specific challenges, we divided the implementation of a cluster into three areas: network, hardware, and software tools. For each area, we performed a study to recognize the trends and obstacles.

The network aspect of clusters has rapidly evolved and is mainly driven by telecommunications. Faster and more efficient communication platforms are always positive, but their implementation in heterogeneous computing has not been obvious, given the existing trade-offs. This part is often fixed in the design process, and its drawbacks are widespread throughout the service stack. From the studied implementations, the following trends and trade-offs were identified:

- Interface: The advantage of a standard interface such as MGTs greatly reduce development efforts at the cost of reducing flexibility. Alternatively, SoC-FPGAs offer numerous GPIOs that can be used at hundreds of megahertz, catching up with the throughput of MGT. The flexibility offered by GPIOs allows the development of custom protocols, such as the time-division multiplexed TDM proposed in [146]. Development time has pushed designers to embrace well-defined interfaces, which are usually constrained by the selected vendor, complicating porting. However, the need for portable interfaces and systems is addressed in [147] by introducing Kyokko, which is a vendor-independent MGT controller.
- Topology: This aspect directly impacts the maximum transmission rate and data throughput in the cluster. In addition, it offers flexibility. Depending on the interface, the topology can be modified in runtime, as in [107] or fixed, as in [64]. For a fixed topology, the 3D torus allows the best physical interconnection at the expense of non-uniform latency if the scaling is not symmetric, as shown in [146]. The alternative of a virtual circuit network over indirect connections shows promise, as shown by VCSN [148] which allows a flexible virtual topology with a performance similar to or better than directly connected networks.
- Protocol: As shown in Table 7, custom protocols remain relevant, suggesting that no industry standard completely satisfies the requirements of heterogeneous computing. This is partially owing to the flexibility of FPGAs, which allows developers to optimize the protocol for latency, as shown in [141]. The drawback is that a more flexible protocol will have a greater complexity, impacting routers, decoders, encoders, and ultimately the network's throughput and latency.

The hardware that constitutes a cluster is another point of discussion. This is closely related to the network, given that the possible interfaces, topologies, and protocols are constrained by the selected hardware. The contribution of Chimera [17] in which they defined the ideal hardware to defeat each of the 13 dwarfs [16] confirms the advantage of heterogeneous nodes, particularly FPGAs, with the inclusion of CPUs or GPUs. This points directly to the SoC-FPGA, which in most cases includes all CEs in a single chip. The main trends identified correspond to an increasing preference for

- CU with fewer CEs
- SoC-FPGAs as the heterogeneous part of the system
- CU standard form factor

Having fewer CEs reduces the CU cost, which is important for the scalability and maintenance of the cluster. A standard form factor facilitates integration in current supercomputing centers by relying on common structural and thermal solutions.

"A tool is only as good as the hands that wield it," is a common saying. In this case, quite often the tool lacks a handle from which to wield it. Software tools are crucial for the usability of any computer, particularly heterogeneous systems that present a new paradigm that makes development much more complicated. This has been the Achilles heel of most implementations, and is one of the reasons that general adoption has yet to occur. We recognize that there are some missing pieces that represent open challenges:

- Hardware abstract model for different heterogeneous platforms.
- Standard interfaces for portability.
- Flexible design tools to optimize implementations targeting heterogeneous clusters.
- Open source tools for community-driven development to further accelerate adoption.
- Operating Systems for cluster management.

**TABLE 9.** Target application and development tools.

| Work Title | Application | Development tools | Isolation level | Flexibility |
|---|---|---|---|---|
| RAW 1997 [32] | Manycore emulator | Custom tools | Programming | None |
| BEE 2003 [117] | DSP emulator | BEE Compiler | Programming, logic | Protocol, external resources |
| BEE2 2005 [120] | RF emulation | BEE Compiler | Programming, logic | Topology |
| COPACOBANA 2006 [53] | Cryptography | Copacobana Lib | Programming | CEs |
| FAST 2006 [33] | Manycore emulator | FAST Software toolbox | User | External resources |
| Janus 2006 [51] | Spin-system simulator | Joslib, Josd, JOS | User | None |
| Maxwell 2007 [88] | General | Parallel ToolKit | User | CEs, topology |
| Spirit 2007 [81] | General |  | None | Protocol, external resources, topology, CEs |
| BEE3 2009 [116] | General | BEE Compiler | Programming, logic | Topology, external resources |
| Cube 2009 [90] | General | Template | Communication | None |
| QP 2009 [91] | General | OFED, Phoenix framework | User | Topology |
| Axel 2010 [94] | General | Custom | Programming | Topology, CEs |
| BEE4 2010 [127] | General | BEE Compiler, Nectar OS, BEE platform studio | Programming, logic | Topology, external resources |
| Formic 2010 [42] | Manycore emulator |  | Programming | Topology, external resources |
| Novo-G 2010 [139] | General | OpenCL, Custom tools | Logic | Topology, CEs |
| RAPTOR 2010 [35] | General | RAPTORLIB, RAPTORAPI, RAPTORGUI | Programming | CEs, external resources |
| Tse, et al. 2010 [57] | Monte-carlo simulations |  | User | Topology, CEs |
| DWCE 2011 [123] | Digital wireless emulator |  | None | None |
| Bluehive 2012 [59] | Neural network simulator | Bluespec | None | Topology, protocol, external resources |
| Chimera 2012 [17] | General |  | Programming | CEs |
| Cuteforce 2013 [56] | Cryptography |  | User | Topology, CEs |
| Catapult 2014 [22] | Bing search engine |  | Logic | Topology |
| Janus II 2014 [63] | Spin-system simulator | Joslib, Josd, JOS | User | None |
| Superdragon 2015 [58] | Cryo-electron microscopy |  | User | Topology, CEs |
| Novo-G# 2016 [98] | General | LEAP OS, Custom tools, Bluespec | Logic | Topology, CEs |
| IBM cloud FPGA 2017 [75] | FaaS |  | Logic | CEs |
| Axiom 2017 [100] | General | OmpSs@FPGA | Programming | Topology, CEs, external resources |
| ARUZ 2018 [25] | General | Custom tools | Logic | None |
| Green Flash 2018 [140] | Astronomic imaging |  | User | Topology, CEs, external resources |
| Noctua 2 2019 [106] | General | Custom tools | Communication | Topology, CEs |
| Astrobyte 2020 [66] | Neural network simulator |  | User | Topology, external resources |
| Enzian 2020 [109] | General | Coyote OS | Logic | Topology, CEs, external resources |
| UNILOGIC 2020 [110] | General | UNILOGIC, Hardware Accelerator Controller | Programming, logic, communication | CEs |
| BiCoSS 2021 [67] | Neural network simulator |  | User | External resources |
| Colosseum 2021 [131] | Digital wireless emulator | BEE Compiler, RFNoC, UHD | Programming, logic | Topology, external resources |
| Cygnus Albeiro 2022 [112] | General | CIRCUS library and API | Programming, logic | CEs |
| ESSPER 2023 [115] | General | R-OPAE, DMA library, AFUShell | Programming, logic | CEs |

- Runtime performance analysis tools to identify bottlenecks.

In 2008, the strategic infrastructure for reconfigurable computing applications (SIRCA) [149] provided a comprehensive study of the tools required for the adoption of mainstream reconfigurable computing. This study separated the tools based on four relevant phases: formulation, design, translation, and execution.

The initial phase, in which the algorithms are elaborated and optimized for parallel computing, is referred to as the formulation. This is the highest level of abstraction, mostly dealing with pseudo-codes and verbal language for reasoning. *SIRCA* highlighted the need for tools that aid developers in making strategic decisions that favor the parallel model embedded in heterogeneous computing rather than leaving the decisions to the later phases. Formulation is the most critical step in which researchers can benefit the most from insight into the paradigm present in the targeted heterogeneous system. Tools that provide strategic exploration, high-level prediction, and numerical analysis have a strongly positive impact on the other phases.

The design phase consists of the languages used to translate an algorithm into a behavioral implementation. This field has been broadened by the creation and adaptation of modern HDL languages, such as Chisel [150], [151] based on Scala and Clash [152], [153] based on Haskell, and high-level synthesis tools, such as BondManchine based on Go [154] among several others. New developments have solved, to some degree, the issues of portability and interoperability by raising abstraction. However, the method for scaling designs to heterogeneous clusters remains platform-specific. Without these facilities, users are expected to be responsible for porting and partitioning the design. Furthermore, users are tasked with specifying the concurrency model at the system level, which is a difficult task. An in-depth study of design tools, frameworks, and strategies for design space exploration can be found in [155].

Once a PC-compatible description of the algorithm is available, the next phase maps it to the actual physical resources of the system. This phase is known as translation or place-and-route (PAR). Several improvements were made in recent years [156], [157]. Most focus on the speed-up of the process by implementing parallelism with good results when compared with vendor tools. However, these improvements are not easily integrated into proprietary workflows and require a high level of expertise for effective usage. Likewise, existing PARs targeting clusters are platform-specific and do not change until a standard way of describing a heterogeneous system is adopted.

In the final phase, execution, developers must be able to verify and analyze the performance of the implementation. Critical runtime services must be included, such as task management, checkpoints, heartbeats, and debugging services. The effective implementation of such services depends on their consideration in previous design phases. The works studied in detail in [158] provided definitions of abstraction layers for user interaction and management, showing great improvement in the execution phase. Likewise, several FPGA operating systems have been developed [159], [160], [161] implementing abstractions such as threads. Nevertheless, some challenges remain, notably:

- Translation tools capable of targeting scalable heterogeneous platforms
- High-level prediction tools for performance, energy consumption, and resource utilization, among others
- Universal debugging and verification tools for distributed reconfigurable computing

Even if there are platform-specific solutions to some of the previously mentioned challenges, the real challenge is to develop standard and generic solutions suitable for any heterogeneous cluster implementation in a community-driven development approach that would greatly accelerate adoption and growth, as shown in [162], [163], and [164]. At high-level synthesis, novel frameworks provide a convenient approach by including off-chip synchronization and communication APIs, such as Auto-Pipe [165] in 2010 and, more recently, OpenFC [166] and SMAPPIC [167].

## IV. CONCLUSION

Supercomputers have been growing in recent years to occupy large areas and consume as much energy as small towns. This trend is impossible to support, and highlights the major issues of the current approach based on CPUs and GPUs. Meanwhile, FPGA-based heterogeneous platforms have shown great improvements in performance and energy consumption when compared to their CPU or GPU counterparts. Nonetheless, adoption has remained low, primarily owing to the complexity of hardware design and the lack of standards for interconnection, structure, and program description, to mention some that affect most development tools by forcing over-specification.

By studying the most relevant implementations of FPGA heterogeneous clusters, we propose three main domains in each cluster, namely, network, hardware, and software tools, that help recognize the contributions and challenges of each work. Furthermore, studying a specific cluster architecture under this division aids in identifying the origin of some issues and understanding the compromises of design decisions taken in the different domains. By understanding the trade-offs related to each decision, developers can better anticipate the critical issues in each domain to plan contingency measures in the most convenient manner. This survey sheds light on the open challenges that future clusters will have to overcome but also offers an overview of the already available and tested approaches.

FPGA-based heterogeneous computing is a challenging field, with enormous potential to change the dominant computing paradigm. In later years, great interest brought important contributions to the development of tools and, more importantly, experimental platforms. With standard platform descriptions and interfaces, an open collaborative development approach will allow the creation of communities to accelerate adoption. New technologies, such as SoC-FPGAs, will certainly be at the center of future cluster architectures, considering the advantages of having CPUs, GPUs, and FPGAs in the same device.

## ACKNOWLEDGMENT

The authors would like to thank Romina Soledad Molina and Charn Loong Ng for their valuable insight in the process of writing this article.

## REFERENCES

[1] C. Maxfield. (Sep. 2011). *Who Made the First PLD?—EETimes*. [Online]. Available: https://www.eetimes.com/who-made-the-first-pld/

[2] (2017). *Xilinx Co-Founder Ross Freeman Honored—EETimes*. [Online]. Available: https://www.eetimes.com/xilinx-co-founder-ross-freeman-honored/

[3] Xilinx. (2021). *Vivado Design Suite User Guide, Version 2021.1*. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_1/ug973-vivado-release-notes-install-license.pdf

[4] Xilinx. (2021). *Vitis Unified Software Platform User Guide, Version 2021.1*. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2021_1/ug1416-vitis-unified-platform.pdf

[5] Intel Corporation. (2021). *Quartus Prime User Guide, Version 21.1*. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qps.pdf

[6] Microsemi Libero. (2021). *Libero SoC Design Suite User Guide, Version 12.0*, 2021. [Online]. Available: https://www.microsemi.com/document-portal/doc_view/131953-libero-soc-design-suite-v12-0-user-guide

[7] (2021). *Yosys Open SYnthesis Suite*. Accessed: May 9, 2023. [Online]. Available: https://github.com/YosysHQ/yosys

[8] CHIPS Alliance. (2017). *FOSS Flows for FPGA—F4PGA Documentation*. [Online]. Available: https://f4pga.readthedocs.io/en/latest/index.html

[9] Agile Analog. (2021). *RapidSilicon: Accelerating Silicon Development*. Accessed: May 9, 2023. [Online]. Available: https://www.agileanalog.com/products/rapidsilicon

[10] W. A. Najjar and P. Ienne, "Reconfigurable computing," *IEEE Micro*, vol. 34, no. 1, pp. 4–6, Jan. 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/508352.508353, doi: 10.1109/MM.2014.25.

[11] Altera Corporation. (Jul. 2014). *What is an SoC FPGA? Architecture Brief*. [Online]. Available: http://www.altera.com/socarchitecture

[12] W. Vanderbauwhede et al., *High-Performance Computing Using FPGAs*. New York, NY, USA: Springer, 2013.

[13] M. Awad, "FPGA supercomputing platforms: A survey," in *Proc. Int. Conf. Field Program. Log. Appl.*, Aug. 2009, pp. 564–568. [Online]. Available: http://ieeexplore.ieee.org/document/5272406/, doi: 10.1109/FPL.2009.5272406.

[14] K. O'Neal and P. Brisk, "Predictive modeling for CPU, GPU, and FPGA performance and power consumption: A survey," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 763–768, doi: 10.1109/ISVLSI.2018.00143.

[15] P. Colella. (2004). Defining Software Requirements for Scientific Computing. DARPA HPCS. [Online]. Available: https://www.krellinst.org/doecsgf/conf/2013/pres/pcolella.pdf

[16] K. Asanovic et al., "The landscape of parallel computing research: A view from Berkeley," 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

[17] R. Inta, D. J. Bowman, and S. M. Scott, "The 'Chimera': An off-the-shelf CPU/GPGPU/FPGA hybrid computing platform," *Int. J. Reconfigurable Comput.*, vol. 2012, pp. 1–10, Jan. 2012. [Online]. Available: http://www.hindawi.com/journals/ijrc/2012/241439, doi: 10.1155/2012/241439.

[18] R. D. Chamberlain, "Architecturally truly diverse systems: A review," *Future Gener. Comput. Syst.*, vol. 110, pp. 33–44, Sep. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X19313184, doi: 10.1016/j.future.2020.03.061.

[19] R. Palmer. (2011). *Parallel Dwarfs (Inaccessible)*. [Online]. Available: http://paralleldwarfs.codeplex.com/

[20] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2009, pp. 44–54, doi: 10.1109/IISWC.2009.5306797.

[21] Virginia Tech Synergy. (2019). *GitHub—VTSynergy/OpenDwarfs: A Benchmark Suite*. [Online]. Available: https://github.com/vtsynergy/OpenDwarfs

[22] A. Putnam et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Jun. 2014, pp. 13–24, doi: 10.1109/ISCA.2014.6853195.

[23] Alibaba. (2018). *Deep Dive Into Alibaba Cloud F3 FPGA as a Service Instances—Alibaba Cloud Community*. [Online]. Available: https://www.alibabacloud.com/blog/deep-dive-into-alibaba-cloud-f3-fpga-as-a-service-instances_594057

[24] Amazon. (2017). *Amazon EC2 F1 Instances*. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/

[25] R. Kiełbik, K. Hałagan, W. Zatorski, J. Jung, J. Ulański, A. Napieralski, K. Rudnicki, P. Amrozik, G. Jabłoński, D. Stożek, P. Polanowski, Z. Mudza, J. Kupis, and P. Panek, "ARUZ—Large-scale, massively parallel FPGA-based analyzer of real complex systems," *Comput. Phys. Commun.*, vol. 232, pp. 22–34, Nov. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010465518302182, doi: 10.1016/j.cpc.2018.06.010.

[26] F. Fahim et al., "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," 2021, *arXiv:2103.05579*.

[27] J. Villarreal, A. Park, W. Najjar, and R. Halstead, "Designing modular hardware accelerators in C with ROCCC 2.0," in *Proc. 18th IEEE Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2010, pp. 127–134, doi: 10.1109/FCCM.2010.28.

[28] R. Nane, V. Sima, B. Olivier, R. Meeuws, Y. Yankova, and K. Bertels, "DWARV 2.0: A CoSy-based C-to-VHDL hardware compiler," in *Proc. 22nd Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2012, pp. 619–622, doi: 10.1109/FPL.2012.6339221.

[29] A. Papakonstantinou, K. Gururaj, J. A. Stratton, D. Chen, J. Cong, and W.-M.-W. Hwu, "Efficient compilation of CUDA kernels for high-performance computing on FPGAs," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2, pp. 1–26, Sep. 2013, doi: 10.1145/2514641.2514652.

[30] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, T. Czajkowski, S. D. Brown, and J. H. Anderson, "LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2, pp. 1–27, Sep. 2013. [Online]. Available: https://doi-org.ezproxy.cern.ch/10.1145/2514740, doi: 10.1145/2514740.

[31] S. Lee, J. Kim, and J. S. Vetter, "OpenACC to FPGA: A framework for directive-based high-performance reconfigurable computing," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 544–554, doi: 10.1109/IPDPS.2016.28.

[32] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, and A. Agarwal, "Baring it all to software: Raw machines," *Computer*, vol. 30, no. 9, pp. 86–93, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/612254/, doi: 10.1109/2.612254.

[33] J. D. Davis, "FAST: A flexible architecture for simulation and testing of multiprocessor and CMP systems," Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, Dec. 2006.

[34] H. Kalte, M. Porrmann, and U. Rückert, "A prototyping platform for dynamically reconfigurable system on chip designs," in *Proc. IEEE Workshop Heterogeneous Reconfigurable Syst. Chip (SoC)*, Hamburg, Germany, Apr. 2002, pp. 57–75.

[35] M. Porrmann et al., "RAPTOR—A scalable platform for rapid prototyping and FPGA-based cluster computing," in *Parallel Computing: From Multicores and GPU's to Petascale* (Advances in Parallel Computing), vol. 19. Amsterdam, The Netherlands: IOS Press, 2010, doi: 10.3233/978-1-60750-530-3-592.

[36] C. Steffen and G. Genest, "Nallatech in-socket FPGA front-side bus accelerator," *Comput. Sci. Eng.*, vol. 12, no. 2, pp. 78–83, Mar. 2010, doi: 10.1109/MCSE.2010.45.

[37] C. Pohl, C. Paiz, and M. Porrmann, "vMAGIC—Automatic code generation for VHDL," *Int. J. Reconfigurable Comput.*, vol. 2009, pp. 1–9, Jan. 2009. [Online]. Available: http://vmagic.sourceforge.net/, doi: 10.1155/2009/205149.

[38] S. Lyberis, G. Kalokerinos, M. Lygerakis, V. Papaefstathiou, I. Mavroidis, M. Katevenis, D. Pnevmatikatos, and D. S. Nikolopoulos, ''FPGA prototyping of emerging manycore architectures for parallel programming research using formic boards,'' *J. Syst. Archit.*, vol. 60, no. 6, pp. 481–493, Jun. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S138376211400054X, doi: 10.1016/j.sysarc.2014.03.002.

[39] S. Lyberis, G. Kalokerinos, M. Lygerakis, V. Papaefstathiou, D. Tsaliagkos, M. Katevenis, D. Pnevmatikatos, and D. Nikolopoulos, ''Formic: Cost-efficient and scalable prototyping of manycore architectures,'' in *Proc. IEEE 20th Int. Symp. Field-Program. Custom Comput. Mach.*, Apr. 2012, pp. 61–64, doi: 10.1109/FCCM.2012.20.

[40] H. Shah et al., ''Remote direct memory access (RDMA) protocol extensions,'' Tech. Rep. 7306, Jun. 2014.

[41] V. Kale, ''Using the MicroBlaze processor to accelerate cost-sensitive embedded system development,'' Xilinx, Jun. 2016. [Online]. Available: https://docs.xilinx.com/v/u/en-US/wp469-microblaze-for-cost-sensitive-apps

[42] S. G. Kavadias, M. G. H. Katevenis, M. Zampetakis, and D. S. Nikolopoulos, ''On-chip communication and synchronization mechanisms with cache-integrated network interfaces,'' in *Proc. 7th ACM Int. Conf. Comput. Frontiers*, May 2010, pp. 217–226, doi: 10.1145/1787275.1787328.

[43] Cadence. (2019). *Palladium Emulation | Cadence*. [Online]. Available: https://www.cadence.com/en_US/home/tools/system-design-and-verification/emulation-and-prototyping/palladium.html

[44] Siemens. (2022). *Veloce Prototyping—FPGA | Siemens Software*. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/veloce/fpga-prototyping/

[45] B. da Silva, A. Braeken, E. H. D'Hollander, A. Touhafi, J. G. Cornelis, and J. Lemeire, ''Comparing and combining GPU and FPGA accelerators in an image processing context,'' in *Proc. 23rd Int. Conf. Field Program. Log. Appl.*, Sep. 2013, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/6645552/, doi: 10.1109/FPL.2013.6645552.

[46] T. Otsuka, T. Aoki, E. Hosoya, and A. Onozawa, ''An image recognition system for multiple video inputs over a multi-FPGA system,'' in *Proc. IEEE 6th Int. Symp. Embedded Multicore SoCs*, Sep. 2012, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/6354671/, doi: 10.1109/MCSoC.2012.33.

[47] The RTN Collaboration, ''64-transputer machine,'' in *Proc. CHEP*, Geneva, Switzerland, 1992, pp. 353–360.

[48] H. Schmit et al., ''Behavioral synthesis for FPGA-based computing,'' in *Proc. IEEE Workshop FPGA's Custom Comput. Mach.*, 1994, pp. 125–132, doi: 10.1109/FPGA.1994.315591.

[49] A. Cruz, J. Pech, A. Tarancón, P. Téllez, C. L. Ullod, and C. Ungil, ''SUE: A special purpose computer for spin glass models,'' *Comput. Phys. Commun.*, vol. 133, nos. 2–3, pp. 165–176, Jan. 2001, doi: 10.1016/S0010-4655(00)00170-3.

[50] F. Belletti, I. Campos, A. Maiorano, S. P. Gavir, D. Sciretti, A. Tarancon, J. L. Velasco, A. C. Flor, D. Navarro, P. Tellez, L. A. Fernandez, V. Martin-Mayor, A. M. Sudupe, S. Jimenez, E. Marinari, F. Mantovani, G. Poll, S. F. Schifano, L. Tripiccione, and J. J. Ruiz-Lorenzo, ''Ianus: An adaptive FPGA computer,'' *Comput. Sci. Eng.*, vol. 8, no. 1, pp. 41–49, Jan. 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1563961/, doi: 10.1109/MCSE.2006.9.

[51] F. Belletti et al., ''Janus: An FPGA-based system for high-performance scientific computing,'' *Comput. Sci. Eng.*, vol. 11, no. 1, pp. 48–58, Jan. 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4720223/, doi: 10.1109/MCSE.2009.11.

[52] M. Baity-Jesi et al., ''An FPGA-based supercomputer for statistical physics: The weird case of Janus,'' in *High-Performance Computing Using FPGAs*. New York, NY, USA: Springer, Mar. 2013, pp. 481–506. [Online]. Available: https://link-springer-com.ezproxy.cern.ch/chapter/10.1007/978-1-4614-1791-0_16, doi: 10.1007/978-1-4614-1791-0_16.

[53] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, ''Breaking ciphers with COPACOBANA—A cost-optimized parallel code breaker,'' in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 4249, 2006, pp. 101–118, doi: 10.1007/11894063_9.

[54] T. Güneysu, T. Kasper, M. Novotný, C. Paar, and A. Rupp, ''Cryptanalysis with COPACOBANA,'' *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1498–1513, Nov. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4515858/, doi: 10.1109/TC.2008.80.

[55] W. Kastl and T. Loimayr, ''A parallel computing system with specialized coprocessors for cryptanalytic algorithms,'' in *P170—Sicherheit 2010—Sicherheit, Schutz und Zuverlässigkeit*, F. C. Freiling, Ed. Bonn, Germany: Gesellschaft für Informatik, 2010, pp. 78–83. [Online]. Available: https://dl.gi.de/handle/20.500.12116/19801

[56] B. Danczul, J. Fuß, S. Gradinger, B. Greslehner, W. Kastl, and F. Wex, ''Cuteforce analyzer: A distributed bruteforce attack on PDF encryption with GPUs and FPGAs,'' in *Proc. Int. Conf. Availability, Rel. Secur.*, Sep. 2013, pp. 720–725. [Online]. Available: http://ieeexplore.ieee.org/document/6657310/, doi: 10.1109/ARES.2013.94.

[57] A. H. T. Tse, D. B. Thomas, K. H. Tsoi, and W. Luk, ''Dynamic scheduling monte-carlo framework for multi-accelerator heterogeneous clusters,'' in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2010, pp. 233–240. [Online]. Available: http://ieeexplore.ieee.org/document/5681495/, doi: 10.1109/FPT.2010.5681495.

[58] G. Tan, C. Zhang, W. Wang, and P. Zhang, ''SuperDragon,'' *ACM Trans. Reconfigurable Technol. Syst.*, vol. 8, no. 4, pp. 1–22, Oct. 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2740966, doi: 10.1145/2740966.

[59] S. W. Moore, P. J. Fox, S. J. T. Marsh, A. T. Markettos, and A. Mujumdar, ''Bluehive–A field-programable custom computing machine for extreme-scale real-time neural network simulation,'' in *Proc. IEEE 20th Int. Symp. Field-Program. Custom Comput. Mach.*, Apr. 2012, pp. 133–140. [Online]. Available: https://ieeexplore.ieee.org/document/6239804/, doi: 10.1109/FCCM.2012.32.

[60] P. J. Fox, A. T. Markettos, and S. W. Moore, ''Reliably prototyping large SoCs using FPGA clusters,'' in *Proc. 9th Int. Symp. Reconfigurable Commun.-Centric Syst.-on-Chip (ReCoSoC)*, May 2014, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/6861350/, doi: 10.1109/ReCoSoC.2014.6861350.

[61] A. Theodore Markettos, P. J. Fox, S. W. Moore, and A. W. Moore, ''Interconnect for commodity FPGA clusters: Standardized or customized?'' in *Proc. 24th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2014, pp. 1–8. http://ieeexplore.ieee.org/document/6927472/, doi: 10.1109/FPL.2014.6927472.

[62] R. S. Nikhil et al., *BSV by Example*, 10th ed. 2010. [Online]. Available: http://www.bluespec.com/support/

[63] M. Baity-Jesi et al., ''Janus II: A new generation application-driven computer for spin-system simulations,'' *Comput. Phys. Commun.*, vol. 185, no. 2, pp. 550–559, Feb. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010465513003470, doi: 10.1016/j.cpc.2013.10.019.

[64] R. Kiełbik, K. Rudnicki, Z. Mudza, and J. Jung, ''Methodology of firmware development for ARUZ—An FPGA-based HPC system,'' *Electronics*, vol. 9, no. 9, p. 1482, Sep. 2020. [Online]. Available: https://www.mdpi.com/journal/electronics, doi: 10.3390/electronics9091482.

[65] (2006). *VHDL Preprocessor Home Page*. [Online]. Available: https://sourceforge.net/projects/vhdlpp/

[66] S. Karim, J. Harkin, L. McDaid, B. Gardiner, and J. Liu, ''AstroByte: Multi-FPGA architecture for accelerated simulations of spiking astrocyte neural networks,'' in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1568–1573, doi: 10.23919/DATE48585.2020.9116312.

[67] S. Yang, J. Wang, X. Hao, H. Li, X. Wei, B. Deng, and K. A. Loparo, ''BiCoSS: Toward large-scale cognition brain with multigranular neuromorphic architecture,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2801–2815, Jul. 2022, doi: 10.1109/TNNLS.2020.3045492.

[68] D. Gratadour. (2021). *Microgate—Green Flash*. [Online]. Available: http://green-flash.lesia.obspm.fr/microgate.html

[69] Y. Clénet et al. (2019). *MICADO-MAORY SCAO Preliminary Design, Development Plan & Calibration Strategies*. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03078430

[70] A. Brown, D. Thomas, J. Reeve, G. Tarawneh, A. De Gennaro, A. Mokhov, M. Naylor, and T. Kazmierski, ''Distributed event-based computing,'' in *Parallel Computing is Everywhere* (Advances in Parallel Computing), vol. 32. 2018, pp. 583–592. [Online]. Available: https://ebooks.iospress.nl/doi/10.3233/978-1-61499-843-3-583, doi: 10.3233/978-1-61499-843-3-583.

[71] M. A. Petrovici, B. Vogginger, P. Müller, O. Breitwieser, M. Lundqvist, L. Müller, M. Ehrlich, A. Destexhe, A. Lansner, R. Schüffny, J. Schemmel, and K. Meier, ''Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms,'' *PLoS ONE*, vol. 9, no. 10, Oct. 2014, Art. no. e108590. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0108590, doi: 10.1371/journal.pone.0108590.

[72] I. Ohmura, G. Morimoto, Y. Ohno, A. Hasegawa, and M. Taiji, "MDGRAPE-4: A special-purpose computer system for molecular dynamics simulations," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 372, Aug. 2014, Art. no. 20130387. [Online]. Available: https://pmc/articles/PMC4084528/ and https://pmc/articles/PMC4084528/?report=abstract and https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4084528/, doi: 10.1098/RSTA.2013.0387.

[73] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Enabling FPGAs in hyperscale data centers," in *Proc. IEEE 12th Int. Conf. Ubiquitous Intell. Comput., IEEE 12th Int. Conf. Autonomic Trusted Comput. IEEE 15th Int. Conf. Scalable Comput. Commun. Associated Workshops (UIC-ATC-ScalCom)*, Aug. 2015, pp. 1078–1086. [Online]. Available: https://ieeexplore.ieee.org/document/7518378/, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.199.

[74] Xilinx. (2016). *Xilinx and IBM to Enable FPGA-Based Acceleration Within SuperVessel OpenPOWER Development Cloud*. [Online]. Available: https://www.xilinx.com/news/press/2016/xilinx-and-ibm-to-enable-fpga-based-acceleration-within-supervessel-openpower-development-cloud.html

[75] F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, and S. Paredes, "An FPGA platform for hyperscalers," in *Proc. IEEE 25th Annu. Symp. High-Perform. Interconnects (HOTI)*, Aug. 2017, pp. 29–32. [Online]. Available: http://ieeexplore.ieee.org/document/8071053/, doi: 10.1109/HOTI.2017.13.

[76] J. Weerasinghe, F. Abel, C. Hagleitner, and A. Herkersdorf, "Disaggregated FPGAs: Network performance comparison against bare-metal servers, virtual machines and Linux containers," in *Proc. Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2016, pp. 9–17, doi: 10.1109/CLOUDCOM.2016.0018.

[77] B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner, and D. Fey, "Programming reconfigurable heterogeneous computing clusters using MPI with transpilation," in *Proc. IEEE/ACM Int. Workshop Heterogeneous High-Perform. Reconfigurable Comput. (H2RC)*, Nov. 2020, pp. 1–9, doi: 10.1109/H2RC51942.2020.00006.

[78] B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner, and D. Fey, "ZRLMPI: A unified programming model for reconfigurable heterogeneous computing clusters," in *Proc. IEEE 28th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2020, p. 220, doi: 10.1109/FCCM48280.2020.00051.

[79] H. Shahzad, A. Sanaullah, and M. Herbordt, "Survey and future trends for FPGA cloud architectures," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2021, pp. 1–10, doi: 10.1109/HPEC49654.2021.9622807.

[80] C. Bobda et al., "The future of FPGA acceleration in datacenters and the cloud," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 3, Sep. 2022, Art. no. 34, doi: 10.1145/3506713.

[81] R. Sass, W. V. Kritikos, A. G. Schmidt, S. Beeravolu, and P. Beeraka, "Reconfigurable computing cluster (RCC) project: Investigating the feasibility of FPGA-based petascale computing," in *Proc. 15th Annu. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2007, pp. 127–140. [Online]. Available: https://ieeexplore.ieee.org/document/4297250, doi: 10.1109/FCCM.2007.62.

[82] A. G. Schmidt, W. V. Kritikos, S. Datta, and R. Sass, "Reconfigurable computing cluster project: Phase I brief," in *Proc. 16th Int. Symp. Field-Program. Custom Comput. Mach.*, Apr. 2008, pp. 300–301, doi: 10.1109/FCCM.2008.12.

[83] AMD Xilinx. (Oct. 2022). *Aurora 64B/66B LogiCORE IP Product Guide*. [Online]. Available: https://docs.xilinx.com/r/en-US/pg074-aurora-64b66b

[84] R. G. Jaganathan, K. D. Underwood, and R. Sass, "A configurable network protocol for cluster based communications using modular hardware primitives on an intelligent NIC," in *Proc. ACM/IEEE Conf. Supercomput.*, Nov. 2003, p. 22, doi: 10.1145/1048935.1050173.

[85] HPC Open. (2022). *Open MPI: Open Source High Performance Computing*. [Online]. Available: https://www.open-mpi.org/

[86] K. Datta and R. Sass, "RBoot: Software infrastructure for a remote FPGA laboratory," in *Proc. 15th Annu. IEEE Symp. Field-Program. Custom Comput. Mach. (FCCM )*, Apr. 2007, pp. 343–344, doi: 10.1109/FCCM.2007.53.

[87] Staff. (Jul. 2005). *FPGA High-Performance Computing Alliance (FHPCA)*. [Online]. Available: http://www.fhpca.org

[88] R. Baxter, S. Booth, M. Bull, G. Cawood, J. Perry, M. Parsons, A. Simpson, A. Trew, A. McCormick, G. Smart, R. Smart, A. Cantle, R. Chamberlain, and G. Genest, "Maxwell—A 64 FPGA supercomputer," in *Proc. 2nd NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Aug. 2007, pp. 287–294. http://ieeexplore.ieee.org/document/4291933/, doi: 10.1109/AHS.2007.71.

[89] R. Baxter, S. Booth, M. Bull, G. Cawood, J. Perry, M. Parsons, A. Simpson, A. Trew, A. McCormick, G. Smart, R. Smart, A. Cantle, R. Chamberlain, and G. Genest, "The FPGA high-performance computing alliance parallel toolkit," in *Proc. 2nd NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Aug. 2007, pp. 301–307, doi: 10.1109/AHS.2007.104.

[90] O. Mencer, K. H. Tsoi, S. Craimer, T. Todman, W. Luk, M. Y. Wong, and P. H. W. Leong, "Cube: A 512-FPGA cluster," in *Proc. 5th Southern Conf. Program. Log. (SPL)*, Apr. 2009, pp. 51–57. [Online]. Available: http://ieeexplore.ieee.org/document/4914907/, doi: 10.1109/SPL.2009.4914907.

[91] M. Showerman, J. Enos, A. Pant, V. Kindratenko, C. Steffen, R. Pennington, and W.-M. Hwu, "QP: A heterogeneous multi-accelerator cluster," in *Proc. 10th LCI Int. Conf. High-Perform. Clustered Comput.*, Boulder, CO, USA, Mar. 2009, pp. 1–8.

[92] Xilinx. (2013). *ISE Design Suite*. [Online]. Available: https://www.xilinx.com/products/design-tools/ise-design-suite.html

[93] A. Pant, H. Jafri, and V. Kindratenko, "Phoenix: A runtime environment for high performance computing on chip multiprocessors," in *Proc. 17th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process.*, 2009, pp. 119–126, doi: 10.1109/PDP.2009.41.

[94] K. H. Tsoi and W. Luk, "Axel," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, New York, NY, USA, Feb. 2010, p. 115. http://portal.acm.org/citation.cfm?doid=1723112.1723134, doi: 10.1145/1723112.1723134.

[95] Adaptive Computing Enterprises. (2015). *TORQUE Resource Manager Administrator Guide 4.2.10*. [Online]. Available: http://www.adaptivecomputing.com

[96] (2014). *Maui Scheduler Administrator's Guide*. [Online]. Available: http://docs.adaptivecomputing.com/maui/

[97] A. George, H. Lam, and G. Stitt, "Novo-G: At the forefront of scalable reconfigurable supercomputing," *Comput. Sci. Eng.*, vol. 13, no. 1, pp. 82–86, Jan. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5678570/, doi: 10.1109/MCSE.2011.11.

[98] A. D. George, M. C. Herbordt, H. Lam, A. G. Lawande, J. Sheng, and C. Yang, "Novo-G#: Large-scale reconfigurable computing with direct and programmable interconnects," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–7. [Online]. Available: http://ieeexplore.ieee.org/document/7761639/, doi: 10.1109/HPEC.2016.7761639.

[99] Xilinx. (Oct. 2017). *Interlaken 150G*. [Online]. Available: https://docs.xilinx.com/v/u/en-US/pg212-interlaken-150g

[100] R. Giorgi, "AXIOM: A 64-bit reconfigurable hardware/software platform for scalable embedded computing," in *Proc. 6th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2017, pp. 1–4. http://ieeexplore.ieee.org/document/7977173/, doi: 10.1109/MECO.2017.7977117.

[101] C. Álvarez et al., "The AXIOM software layers," *Microprocessors Microsyst.*, vol. 47, pp. 262–277, Nov. 2016, doi: 10.1016/J.MICPRO.2016.07.002.

[102] R. Giorgi, M. Procaccini, and F. Khalili, "AXIOM: A scalable, efficient and reconfigurable embedded platform," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 480–485, doi: 10.23919/DATE.2019.8715168.

[103] A. Filgueras, M. Vidal, M. Mateu, D. Jiménez-González, C. Alvarez, X. Martorell, E. Ayguadé, D. Theodoropoulos, D. Pnevmatikatos, P. Gai, S. Garzarella, D. Oro, J. Hernando, N. Bettin, A. Pomella, M. Procaccini, and R. Giorgi, "The AXIOM project: IoT on heterogeneous embedded platforms," *IEEE Design Test*, vol. 38, no. 5, pp. 74–81, Oct. 2021, doi: 10.1109/MDAT.2019.2952335.

[104] AMD-Xilinx. (2021). *Xilinx Adaptive Compute Clusters (XACC) Academia-Industry Research Ecosystem | HACC Resources*. [Online]. Available: https://www.amd-haccs.io/adapt_2021.html

[105] (2016). *Heterogeneous Accelerated Compute Clusters | HACC Resources*. [Online]. Available: https://www.amd-haccs.io/index.html

[106] T. Prickett. (2018). *Forging a Hybrid CPU-FPGA Supercomputer*. [Online]. Available: https://www.nextplatform.com/2018/09/25/forging-a-hybrid-cpu-fpga-supercomputer/

[107] Paderborn Center for Parallel Computing (PC2). (2022). *PC2—Noctua 2 (Universität Paderborn)*. [Online]. Available: https://pc2.uni-paderborn.de/hpc-services/available-systems/noctua2

[108] Intel. (2022). *OneAPI: A New Era of Heterogeneous Computing*. [Online]. Available: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html

[109] D. Cock, A. Ramdas, D. Schwyn, M. Giardino, A. Turowski, Z. He, N. Hossle, D. Korolija, M. Licciardello, K. Martsenko, R. Achermann, G. Alonso, and T. Roscoe, "Enzian: An open, general, CPU/FPGA platform for systems software research," in *Proc. 27th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Feb. 2022, p. 18, doi: 10.1145/3503222.3507742.

[110] A. D. Ioannou, K. Georgopoulos, P. Malakonakis, D. N. Pnevmatikatos, V. D. Papaefstathiou, I. Papaefstathiou, and I. Mavroidis, "UNILOGIC: A novel architecture for highly parallel reconfigurable systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, no. 4, pp. 1–32, Dec. 2020, doi: 10.1145/3409115.

[111] Cygnus Consortium. (2018). *About Cygnus*. [Online]. Available: https://www.ccs.tsukuba.ac.jp/wp-content/uploads/sites/14/2018/12/About-Cygnus.pdf

[112] T. Boku, N. Fujita, R. Kobayashi, and O. Tatebe, "Cygnus—World first multihybrid accelerated cluster with GPU and FPGA coupling," in *Proc. ICPP Workshops*. New York, NY, USA: Association for Computing Machinery, Aug. 2022, p. 1, doi: 10.1145/3547276.3548629.

[113] K. Kikuchi, N. Fujita, R. Kobayashi, and T. Boku, "Implementation and performance evaluation of collective communications using CIRCUS on multiple FPGAs," in *Proc. HPC Asia Workshops*. New York, NY, USA: Association for Computing Machinery, Feb. 2023, p. 1523, doi: 10.1145/3581576.3581602.

[114] RIKEN Center for Computational Science. (2020). *Fugaku: Riken's Flagship Supercomputer*. [Online]. Available: https://www.fugaku-riken.jp/

[115] K. Sano, A. Koshiba, T. Miyajima, and T. Ueno, "ESSPER: Elastic and scalable FPGA-cluster system for high-performance reconfigurable computing with supercomputer Fugaku," in *Proc. Int. Conf. High Perform. Comput. Asia–Pacific Region (HPC Asia)*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 140–150, doi: 10.1145/3578178.3579341.

[116] J. Davis et al., "BEE3: Revitalizing computer architecture research," Microsoft, Apr. 2009. [Online]. Available: https://www.microsoft.com/en-us/research/publication/bee3-revitalizing-computer-architecture-research/

[117] K. Kuusilinna, C. Chang, M. J. Ammer, B. C. Richards, and R. W. Brodersen, "Designing BEE: A hardware emulation engine for signal processing in low-power wireless applications," *EURASIP J. Adv. Signal Process.*, vol. 2003, no. 6, pp. 502–513, Dec. 2003. [Online]. Available: https://www.mathworks.com

[118] S. C. Jain, S. Kumar, and A. Kumar, "Evaluation of various routing architectures for multi-FPGA boards," in *Proc. VLSI Design Wireless Digit. Imag. Millennium 13th Int. Conf. VLSI Design*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 262–267. [Online]. Available: http://ieeexplore.ieee.org/document/812619/, doi: 10.1109/ICVD.2000.812619.

[119] C. Chang, K. Kuusilinna, B. Richards, and R. W. Brodersen, "Implementation of BEE: A real-time large-scale hardware emulation engine," in *Proc. ACM/SIGDA 11th Int. Symp. Field Program. Gate Arrays*, Feb. 2003, pp. 91–99, doi: 10.1145/611817.611832.

[120] C. Chang, J. Wawrzynek, and R. W. Brodersen, "BEE2: A high-end reconfigurable computing system," *IEEE Design Test Comput.*, vol. 22, no. 2, pp. 114–125, Feb. 2005, doi: 10.1109/MDT.2005.30.

[121] A. G. Schmidt, B. Huang, R. Sass, and M. French, "Checkpoint/restart and beyond: Resilient high performance computing with FPGAs," in *Proc. IEEE 19th Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2011, pp. 162–169, doi: 10.1109/FCCM.2011.22.

[122] S. Buscemi and R. Sass, "Design and utilization of an FPGA cluster to implement a digital wireless channel emulator," in *Proc. 22nd Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2012, pp. 635–638, doi: 10.1109/FPL.2012.6339253.

[123] S. Buscemi and R. Sass, "Design of a scalable digital wireless channel emulator for networking radios," in *Proc. Mil. Commun. Conf.*, Nov. 2011, pp. 1858–1863. [Online]. Available: http://ieeexplore.ieee.org/document/6127583/, doi: 10.1109/MILCOM.2011.6127583.

[124] D. A. Patterson, "RAMP: Research accelerator for multiple processors—A community vision for a shared experimental parallel HW/SW platform," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, Mar. 2006, p. 1, doi: 10.1109/ISPASS.2006.1620784.

[125] Wirbel Loring. (May 2010). *Berkeley Emulation Engine Update—EDN*. [Online]. Available: https://www.edn.com/berkeley-emulation-engine-update/

[126] J. Rothman and C. Chang, "BEE technology overview," in *Proc. Int. Conf. Embedded Comput. Syst. (SAMOS)*. Samos, Greece: Institute of Electrical and Electronics Engineers, Jan. 2013, p. 277, doi: 10.1109/SAMOS.2012.6404186.

[127] EDN. (Jun. 2010). *DESIGN TOOLS—BEEcube Launches BEE4, a Full-Speed FPGA Prototyping Platform—EDN*. [Online]. Available: https://www.edn.com/design-tools-beecube-launches-bee4-a-full-speed-fpga-prototyping-platform/

[128] M. Lin, "Hardware-assisted large-scale neuroevolution for multiagent learning," Dept. Elect. Comput. Eng., Univ. Central Florida, Orlando, FL, USA, Dec. 2014. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA621804

[129] I. Sokol. (Apr. 2015). *NIs BEEcube Acquisition Drives 5G Communications | Microwaves & RF*. [Online]. Available: https://www.mwrf.com/technologies/systems/article/21846169/nis-beecube-acquisition-drives-5g-communications

[130] National Instruments. (2022). *What is FlexRIO?—NI*. [Online]. Available: https://www.ni.com/it-it/shop/electronic-test-instrumentation/flexrio/what-is-flexrio.html

[131] L. Bonati, P. Johari, M. Polese, S. D'Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltser, F. Restuccia, A. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, "Colosseum: Large-scale wireless experimentation through hardware-in-the-loop network emulation," in *Proc. IEEE Int. Symp. Dyn. Spectr. Access Netw. (DySPAN)*, Dec. 2021, pp. 105–113, doi: 10.1109/DYSPAN53946.2021.9677430.

[132] Ettus. (2014). *USRP Hardware Driver and USRP Manual: USRP X3x0 Series*. [Online]. Available: https://files.ettus.com/manual/page_usrp_x3x0.html

[133] NI. (2022). *ATCA Overview—NI*. [Online]. Available: https://www.ni.com/docs/en-US/bundle/atca-3671-getting-started/page/overview.html

[134] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: Topology, routing, and packaging of efficient large-scale networks," in *Proc. Conf. High Perform. Comput. Netw., Storage Anal. (SC)*, New York, NY, USA, 2009, p. 1. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1654059.1654101, doi: 10.1145/1654059.1654101.

[135] S. Gupta et al. (2022). *Getting Started With RFNoC in UHD 4.0—Ettus Knowledge Base*. [Online]. Available: https://kb.ettus.com/Getting_Started_with_RFNoC_in_UHD_4.0

[136] A. Chaudhari and M. Braun, "A scalable FPGA architecture for flexible, large-scale, real-time RF channel emulation," in *Proc. 13th Int. Symp. Reconfigurable Commun.-Centric Syst.-on-Chip (ReCoSoC)*, Jul. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8449390/, doi: 10.1109/ReCoSoC.2018.8449390.

[137] J. J. Dongarra and A. J. van der Steen, "High-performance computing systems: Status and outlook," *Acta Numerica*, vol. 21, pp. 379–474, May 2012, doi: 10.1017/S0962492912000050.

[138] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. M. Elfadel, "FPGAaaS: A survey of infrastructures and systems," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 1143–1156, Mar. 2022, doi: 10.1109/TSC.2020.2976012.

[139] A. George, H. Lam, A. Lawande, C. Pascoe, and G. Stitt, "Novo-G: A view at the HPC crossroads for scientific computing," in *Proc. ERSA*, 2010, pp. 21–30. [Online]. Available: http://plaza.ufl.edu/poppyc/ERS5029.pdf

[140] D. Gratadour et al., "Prototyping AO RTC using emerging high performance computing technologies with the green flash project," *Proc. SPIE*, vol. 10703, pp. 404–418, Jul. 2018. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10703/1070318/Prototyping-AO-RTC-using-emerging-high-performance-computing-technologies-with/10.1117/12.2312686.full%20, doi: 10.1117/12.2312686.

[141] A. Mondigo, T. Ueno, K. Sano, and H. Takizawa, "Comparison of direct and indirect networks for high-performance FPGA clusters," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications* (Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12083. Springer, 2020, pp. 314–329. [Online]. Available: http://link.springer.com/10.1007/978-3-030-44534-8_24, doi: 10.1007/978-3-030-44534-8_24.

[142] J. D. D. Gazzano, M. L. Crespo, A. Cicuttin, and F. R. Calle, *Field-Programmable Gate Array (FPGA) Technologies for High Performance Instrumentation*. Hershey, PA, USA: IGI Global, Jul. 2016, doi: 10.4018/978-1-5225-0299-9.

[143] J. P. Orellana, M. B. Caminero, and C. Carrión, "Diseño de una arquitectura heterogénea para la gestión eficiente de recursos FPGA en un cloud privado," in *Aplicaciones e Innovación de la Ingeniería en Ciencia y Tecnología*. Quito, Ecuador: Abya-Yala, 2019, pp. 165–199, doi: 10.7476/9789978104910.0007.

[144] M. Southworth. (Oct. 2021). Choosing the best processor for the job. Curtis-Wright. [Online]. Available: https://www.curtisswrightds.com/sites/default/files/2021-10/Choosing-the-Best-Processor-for-the-Job-white-paper.pdf

[145] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels," in *Proc. IEEE Int. Conf. Embedded Softw. Syst. (ICESS)*, Jun. 2019, pp. 1–8, doi: 10.1109/ICESS.2019.8782524.

[146] A. Cicuttin, M. L. Crespo, K. S. Mannatunga, J. G. Samarawickrama, N. Abdallah, and P. B. Sabet, "HyperFPGA: A possible general purpose reconfigurable hardware for custom supercomputing," in *Proc. Int. Conf. Adv. Electr., Electron. Syst. Eng. (ICAEES)*, Nov. 2016, pp. 21–26, doi: 10.1109/ICAEES.2016.7888002.

[147] A. Tomori and Y. Osana, "Kyokko: A vendor-independent high-speed serial communication controller," in *Proc. 11th Int. Symp. Highly Efficient Accel. Reconfigurable Technol.* New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 1–6. [Online]. Available: https://doi-org.ezproxy.cern.ch/10.1145/3468044.3468051, doi: 10.1145/3468044.3468051.

[148] T. Ueno and K. Sano, "VCSN: Virtual circuit-switching network for flexible and simple-to-operate communication in HPC FPGA cluster," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 16, no. 2, pp. 1–32, Jun. 2023, doi: 10.1145/3579848.

[149] T. El-Ghazawi et al., "Exploration of a research roadmap for application development and execution on field-programmable gate array (FPGA)-based systems," George Washington Univ., Washington, DC, USA, Tech. Rep. ADA494473, Oct. 2008. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA494473

[150] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanovic, "Chisel: Constructing hardware in a scala embedded language," in *Proc. Design Autom. Conf.*, 2012, pp. 1216–1225, doi: 10.1145/2228360.2228584.

[151] A. Izraelevitz, J. Koenig, P. Li, R. Lin, A. Wang, A. Magyar, D. Kim, C. Schmidt, C. Markley, J. Lawson, and J. Bachrach, "Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations," in *IEEE/ACM Int. Conf. Comput.-Aided Design Dig. Tech. Papers*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Nov. 2017, pp. 209–216, doi: 10.1109/ICCAD.2017.8203780.

[152] C. Baaij, "CλasH: From Haskell to hardware," Fac. EEMCS. Comput. Archit. Embedded Syst., Univ. Twente, Enschede, The Netherlands, Dec. 2009.

[153] M. Kooijman, "Haskell as a higher order structural hardware description language," Fac. EEMCS, Comput. Archit. Embedded Syst., Univ. Twente, Enschede, The Netherlands, Dec. 2009. [Online]. Available: http://essay.utwente.nl/59381/

[154] M. Mariotti, D. Magalotti, D. Spiga, and L. Storchi, "The bondmachine, a moldable computer architecture," *Parallel Comput.*, vol. 109, Mar. 2022, Art. no. 102873, doi: 10.1016/J.PARCO.2021.102873.

[155] R. S. Molina, V. Gil-Costa, M. L. Crespo, and G. Ramponi, "High-level synthesis hardware design for FPGA-based accelerators: Models, methodologies, and frameworks," *IEEE Access*, vol. 10, pp. 90429–90455, 2022, doi: 10.1109/ACCESS.2022.3201107.

[156] Y. Zhou, D. Vercruyce, and D. Stroobandt, "Accelerating FPGA routing through algorithmic enhancements and connection-aware parallelization," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, no. 4, pp. 1–26, Dec. 2020, doi: 10.1145/3406959.

[157] M. A. Zapletina and D. A. Zheleznikov, "The acceleration techniques for the modified pathfinder routing algorithm on an island-style FPGA," in *Proc. Conf. Russian Young Res. Electr. Electron. Eng. (ElConRus)*, Jan. 2022, pp. 920–923. [Online]. Available: https://ieeexplore.ieee.org/document/9755536/, doi: 10.1109/ElConRus54750.2022.9755536.

[158] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on FPGA virtualization," in *Proc. 28th Int. Conf. Field Program. Log. Appl. (FPL)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Aug. 2018, pp. 131–138, doi: 10.1109/FPL.2018.00031.

[159] K. Fleming, H. Yang, M. Adler, and J. Emer, "The LEAP FPGA operating system," in *Proc. 24th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2014, pp. 1–8, doi: 10.1109/FPL.2014.6927488.

[160] L. Clausing and M. Platzner, "ReconOS64: A hardware operating system for modern platform FPGAs with 64-bit support," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2022, pp. 120–127, doi: 10.1109/IPDPSW55747.2022.00029.

[161] D. Korolija, T. Roscoe, and G. Alonso, "Do OS abstractions make sense on FPGAs?" in *Proc. 14th USENIX Symp. Operating Syst. Design Implement.*, 2020, pp. 991–1010. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/roscoe, doi: 10.5555/3488766.3488822.

[162] S. Möller et al., "Community-driven development for computational biology at sprints, hackathons and codefests," *BMC Bioinf.*, vol. 15, Dec. 2014, Art. no. S7, doi: 10.1186/1471-2105-15-S14-S7.

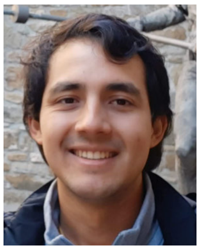
[163] M. Pathan et al., "A novel community driven software for functional enrichment analysis of extracellular vesicles data," *J. Extracellular Vesicles*, vol. 6, no. 1, Dec. 2017, Art. no. 1321455, doi: 10.1080/20013078.2017.1321455.

[164] M. Kühbach, A. J. London, J. Wang, D. K. Schreiber, F. M. Martin, I. Ghamarian, H. Bilal, and A. V. Ceguerra, "Community-driven methods for open and reproducible software tools for analyzing datasets from atom probe microscopy," *Microsc. Microanal.*, vol. 28, no. 4, pp. 1038–1053, Aug. 2022. [Online]. Available: https://www.cambridge.org/core/product/identifier/S1431927621012241/type/journal_article, doi: 10.1017/S1431927621012241.

[165] R. D. Chamberlain, M. A. Franklin, E. J. Tyson, J. H. Buckley, J. Buhler, G. Galloway, S. Gayen, M. Hall, E. F. B. Shands, and N. Singla, "Auto-pipe: Streaming applications on architecturally diverse systems," *Computer*, vol. 43, no. 3, pp. 42–49, Mar. 2010, doi: 10.1109/MC.2010.62.

[166] Y. Osana, T. Imahigashi, and A. Tomori, "OpenFC: A portable toolkit for custom FPGA accelerators and clusters," in *Proc. 8th Int. Symp. Comput. Netw. Workshops (CANDARW)*, Nov. 2020, pp. 185–190, doi: 10.1109/CANDARW51189.2020.00045.

[167] G. Chirkov and D. Wentzlaff, "SMAPPIC: Scalable multi-FPGA architecture prototype platform in the cloud," in *Proc. 28th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.* New York, NY, USA: Association for Computing Machinery, Jan. 2023, pp. 733–746, doi: 10.1145/3575693.3575753.



**WERNER FLORIAN SAMAYOA** received the B.S. degree in electronics engineering from the University of San Carlos, Guatemala, in 2018. He is currently pursuing the Ph.D. degree in industrial and information engineering with the Multidisciplinary Laboratory (MLab), The Abdus Salam International Center for Theoretical Physics, Universitã degli Studi di Trieste, under the Joint-Supervision Program. His research interest includes scalable reconfigurable supercomputing.

**MARIA LIZ CRESPO** is currently a Research Officer with The Abdus Salam International Centre for Theoretical Physics (ICTP) and an Associate Researcher with the Italian National Institute of Nuclear Physics (INFN), Trieste, Italy. She is also coordinating the Research and Training Program, Multidisciplinary Laboratory (MLab), ICTP. She has organized several international schools and workshops on fully programmable systems on chip for nuclear and scientific instrumentation. She is the coauthor of more than 100 scientific publications in prestigious peer-reviewed journals. Her research interests include advanced scientific instrumentation for particle physics experiments and experimental multidisciplinary research.

**ANDRES CICUTTIN** received the degree in physics from the National University of La Plata, Argentina, in 1992, and the Laurea degree in fisica from the University of Trieste, Italy, in 1993. He is currently a Technical Assistant with the Multidisciplinary Laboratory, The Abdus Salam International Centre for Theoretical Physics, and an Associate Researcher with the Italian National Institute for Nuclear Physics (INFN). He has organized and directed numerous international workshops on programmable logic devices for scientific instrumentation and high education.

**SERGIO CARRATO** received the master's degree in electronic engineering and the Ph.D. degree in signal processing from the University of Trieste, Trieste, Italy. Then, he was with Ansaldo Componenti and Sincrotrone Trieste in the field of electronic instrumentation for applied physics. He joined the Department of Electronics, University of Trieste, where he is currently an Associate Professor in electronic devices.

Open Access funding provided by 'Università degli Studi di Trieste' within the CRUI CARE Agreement