Implementation and impact of an ultra-compact multi-FPGA board for large system prototyping

F. Chaix, A.D. Ioannou*, N. Kossifidis, N. Dimou, G. Ieronymakis, M. Marazakis, V. Papaefstathiou,
V. Flouris, M. Ligerakis, G. Ailamakis, T.C. Vavouris, A. Damianakis, M. G.H. Katevenis, I. Mavroidis
Foundation for Research and Technology - Hellas (FORTH), Heraklion, Greece

***Abstract*—Efficient prototyping of a large complex system can be significantly facilitated by the use of a flexible and versatile physical platform where both new hardware and software components can readily be implemented and tightly integrated in a timely manner. Towards this end, we have developed the 120×130 mm Quad-FPGA Daughter Board (QFDB) and associated firmware, including the system software environment. We developed a large system based on this advanced dense and modular building block. The QFDB features 4 interconnected Xilinx Zynq Ultrascale+ devices, each one consisting of an ARM-based subsystem tightly coupled with reconfigurable logic. Each Zynq Ultrascale+ is connected to 16 GB of DDR4 memory. In addition, one Zynq provides storage through an M.2 Solid State Disk (SSD). In this paper, we present the design and the implementation of this board, as well as the software environment for board operation. Moreover, we describe a 10 Gb Ethernet communication infrastructure for interconnecting multiple boards together. Finally, we highlight the impact of this board on a number of ongoing research activities that leverage the QFDB versatility, both as a large-scale prototyping system for HPC solutions, and as a host for the development of FPGA integration techniques.**

***Index Terms*—Prototyping, Accelerator architectures, Large scale systems, Emulation, ARM multiprocessor, System on Module, Field Programmable Gate Arrays, High Performance Computing, Transceivers, High speed connectivity**

I. Introduction

The design and prototyping of next-generation high performance computing systems demands the capability to develop and experiment with research features at a significant scale. Unfortunately, the implementation of many techniques is often impractical or impossible in vendor systems due to performance, cost, or complexity issues. Within the ExaNeSt H2020 project [1], we take advantage of the reconfigurable resources and the high-speed connectivity of modern FPGA-based System-on-Chips (SoCs) to design a high-density development board that serves as a building block for prototyping such large-scale systems.

The Quad-FPGA Daughter Board (QFDB) provides a more compact and advanced solution than prior similar development platforms in multiprocessor emulators and networking such as the RPM [2], the BEE2 [3], the RAMP [4], and the NetFPGA [5]. The main design goal of these FPGA-based systems has been to support the development of new hardware/software solutions, offering full observability of the status at any point in the system. The capability to probe the status of a large-scale system, trigger events, and execute complex actions on such triggers, is also available in software simulators, but with the inevitable software delays [6].

We present the QFDB, a high performance, energy-efficient System-On-Module, supporting four state-of-the-art Xilinx ZU9EG FPGAs. Each FPGA contains a quad-core ARM A53 processor, 600K reconfigurable logic cells, 2520 DSP slices, and 32 Mb of internal memory. In addition, the module offers 16 GB of DDR4 memory connected to each FPGA and an M.2 SSD. In this paper, the architecture and the implementation of the QFDB are presented, including technical issues and important experience gained. Moreover, we describe a communication infrastructure for interconnecting multiple QFDB boards that accommodates various experimental requirements.

Finally, we briefly describe a number of achievements that build upon the QFDB features. An ad-hoc high-speed low-latency interconnect and its software stack were developed [7]. In addition, a high-density rack was assembled to showcase the potential of the ExaNeSt technology. QFDBs were also utilized to develop a framework for the utilization of hardware accelerators [8], [9].

Overall, the QFDB has been extensively used by four European HPC projects, which strongly collaborated to develop technical solutions for future HPC systems [1], [10]–[12]. Currently, we have developed a HPC prototype that supports 48 interconnected boards, amounting to 192 Zynq Ultrascale+, 3 TB of DDR4 memory and 12 TB of non-volatile storage.

II. Related work

In the last decade, there has been an increasing interest on multi-FPGA systems. The Convey-HC1 [13] and the RAMP accelerator [4] were popular such platforms. These systems are usually used as hardware accelerators or as hardware emulators for fast prototyping.

Most of todays FPGA-based boards that can be employed in larger systems, such as Bittware [14], Hitech Global [15] and Digilent [16] rely on Intel processors connected to FPGA boards over PCIe, which can impact the communication latency. A larger PCIe-based hardware accelerator is the Novo-G# system [17] that integrates 192 Stratix IV, and focuses on communication-intensive applications.

There are several other multi-FPGA systems. For instance, the Amazon EC2 F1 instances contain up to 8 FPGAs [18], the Maxeler MPC nodes accelerate applications on reconfigurable

*also at the School of ECE, Technical University of Crete, Chania, Greece

978-1-7281-5999-7/19/$31.00 ©2019 IEEE
DOI 10.1109/H2RC49586.2019.00010

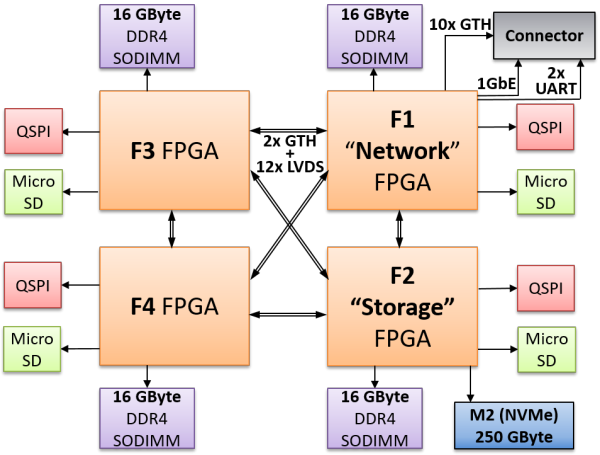
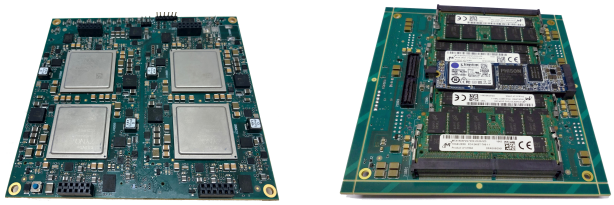


Fig. 1: Block Diagram of the QFDB board



(a) Top view (b) Bottom view

Fig. 2: Pictures of the QFDB

Data Flow Engines (DFEs) [19], [20], and Rivyera from SciEngines that support up to 128 Xilinx Spartan FPGAs per machine [21]. Moreover, there are large hardware FPGA-based emulators such as Mentor's Veloce [22] and Cadence Palladium Z1 [23]. These approaches are based on large monolithic development systems, while the QFDB provides a compact self-contained subsystem that can be used in a modular architecture to build large systems supporting a wider variety of designs.

IBM proposes a dense multi-FPGA platform [24] that targets the Data Center, where FPGAs are completely decoupled from CPUs. QFDBs attain the same density but retain ARM cores tightly attached to the Programmable Logic.

Finally, there are similar approaches to the QFDB, which provide tightly coupled reconfigurable resources, such as BEE7 from BEEcube [25] and SG280 from ProFPGA [26]. However, the QFDB module provides more capabilities (DDR4 memory, storage, monitoring) on a much denser board (about $\times 5$ to $\times 10$ smaller). It is thus built focusing on flexibility and scalability, without missing any feature.

III. Board Architecture

As shown in Figure 1, the QFDB 4 Xilinx Zynq Ultrascale+ devices (ZU9EG). Each device features a Processing System (PS) that consists of 4 ARM-A53 and 2 ARM-R5 cores along with a rich set of hard IPs, and reconfigurable logic resources also referred as Programmable Logic (PL). A 16 GB DDR4 SO-DIMM and a 32 MB QSPI memory are connected to each Zynq device. In addition, a 250 GB M.2 SSD is connected to the "Storage" FPGA, while 2 TB devices are available today [27]. The power dissipated during normal operation is usually close to 50 Watts, but it can reach up to 120 Watts when using the Stress IP presented in Section IV-A.

In order to meet the constraints of the system described in Section VI-B, the Printed Circuit Board (PCB) dimensions were narrowed to 120×130 mm and the height of all components on top and bottom sides was kept under 10 mm, as shown in Figure 2. The PCB stack-up consists of 16 layers using Megtron-6 dielectric. Table I details the amount of the high-speed PCB traces that had to be routed. The high concentration of components and high-speed traces (almost a thousand) required significant effort for the placement of the components and the routing of the traces. As a side effect, it was impossible to include either any BMC or CPLD to control the power sequence or configuration signals, thus impeding the bring-up process. The power tree is using 48 V input, which is immediately transformed into 12 V, and from there to adequate supply voltages, totaling 30 regulators.

There is also a variety of communication paths on this board, as shown in Figure 1. One Zynq is connected to the outside world through 10 High Speed Serial Links (HSSL) by means of GTH transceivers. Each external link has a maximum rate of 10.3125 Gb/s. The MAC-to-PHY RGMII interface also allows connection to 1 Gb Ethernet (GbE) for management purposes. Within the board, each Zynq is connected to each other through both 2 HSSL capable of operating at the maximum lane rate of 16.375 Gb/s, and 24 LVDS pairs (12 in each direction). Finally, the board supports as much as 15 I2C power sensors, which allows the monitoring of the main sub-systems of the board.

Running DGEMM [28], we measured that the ARM quad-core processor of a single device can execute up to 7.9 GFLOP/s. When running the STREAM program [29], the DDR subsystem of the device sustained a maximum throughput of respectively 6488 MB/s for Copy, 5886 MB/s for Scale, 4269 MB/s for Add, and 4032 MB/s for Triad. Finally, using the Fio [30] benchmark, we measured that the sequential device read throughput of the SSD is 1.3 GB/s.

IV. Board Implementation Details

In this section, important aspects of the QFDB implementation and of the software environment are discussed, offering insight for the development of such complex and dense compute nodes.

TABLE I: High speed signals in the QFDB

Type	Count	Maximum Speed
DDR4	532	1200 MHz / 2400 Mb/s
LVDS	288	800 MHz / 1600 Mb/s
HSSL	88	16.375 Gb/s
PCIe	16	5 Gb/s

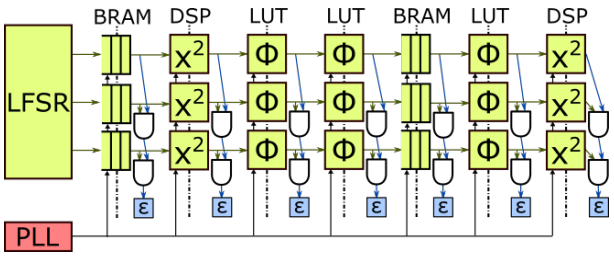


Fig. 3: Diagram of the stress IP

A. Stress Test and Monitoring

Once a functional QFDB was built, the limits of the board were probed. The first action was to develop a monitoring infrastructure to ensure that boards are kept in safe conditions at all times. The second task has been to develop a stress test IP that is able to push the Programmable Logic of any FPGA to its limits in a controllable fashion, and quickly give a view of its maximum computing and electric power. The Processing System of the Zynq was stressed as well by running diverse high-performance applications.

Our monitoring tool continuously monitors any sensor present on the QFDB, independently of the software running on the Zynq, continuously display a condensed view that highlights the criticality of each value with respect to predefined allowed operation conditions. The complete data set is also saved to help with postmortem investigations.

Then, the capabilities of the Programmable Logic (PL) part of the Zynq were reviewed extensively. In effect, depending on the firmware in use, the PL of each Zynq may consume from 2 to more than 16 Watts. Hence, the board might work with light firmware but be deficient with heavier firmware. Figure 3 provides an overview of the stress IP, which essentially consists of permutations (Φ) to exercise Lookup Tables (LUTs), arithmetic operations to exercise the hard DSP blocks, and FIFOs to exercise the Block RAMs (BRAM). These circuits are fed by a Linear Feedback Shift Register (LFSR) to reach both adequate coverage and high activity rate. The IP structure follows the physical distribution of the PL. For each column of elements, errors are monitored to detect noise and voltage issues (ϵ). The whole IP is fed by a PLL hard block that allows us to increase progressively the operating frequency. This offers the possibility to gradually increase load intensity and hence fully control temperature and timely detect side effects.

B. Bring-up and deployment aspects

The actual implementation of a board as complex and compact as the QFDB often comes with many setbacks. Hence, we discuss here a few aspects of the bring-up that can apply to many other boards. In addition, coping with the large prototype size demanded that we prepare early for the deployment of tens of boards. An exhaustive factory test was developed, and an identification scheme was devised in order to keep track of the boards and ease the design of a robust MAC addressing scheme.

The high complexity and density of the QFDB impeded its bring-up, and required a variety of advanced techniques. In effect, due to our stringent density constraints, there was little headroom on the PCB to aid with hardware debug. At first, a board without any FPGA was used to patch and validate the power tree. Then, partially populated boards have been used to validate active components. Once the JTAG chain was validated, we carried on with the bring-up of a first set of functionalities. A set of debug bitstreams was also generated to validate the functionalities pertaining to the Programmable Logic of the Zynqs. Finally, a Linux OS instance was booted on each MPSoC, to validate the Ethernet interface and the SSD storage.

Then, the routines and firmware developed during the bring-up phase were packed together to systematically prepare new boards before their deployment in the prototype. Initially, each sub-system is tested. Then, the QSPI memories are flashed with an initial production image, and the Linux environment presented in Section V-A is booted on each Zynq of the board. Finally, each Zynq is provisioned. The Zynq Ultrascale+ eFUSEs are used to permanently keep these data. Our eFUSEs structure contains a unique 64-bit Board ID, the node ID within the board, the board serial number and revision, along with information about specific board capabilities.

C. Memory Borrowing and RAM-less Linux boot

During the bring-up of the QFDB, a memory-borrowing environment was developed to speed up the validation of the board. In this setup, presented in Figure 4, the Linux OS was booted by borrowing the memory of a third-party "donor" board. This technique may be applied to any other FPGA-based board lacking significant external memory. In our case, a Trenz TEBF0808 [31] was used, which features an identical Zynq Ultrascale+ FPGA along with 2 GB of DDR4-RAM.

Booting a Linux in this setup presented many pitfalls. On the software side, a very minimal software environment was

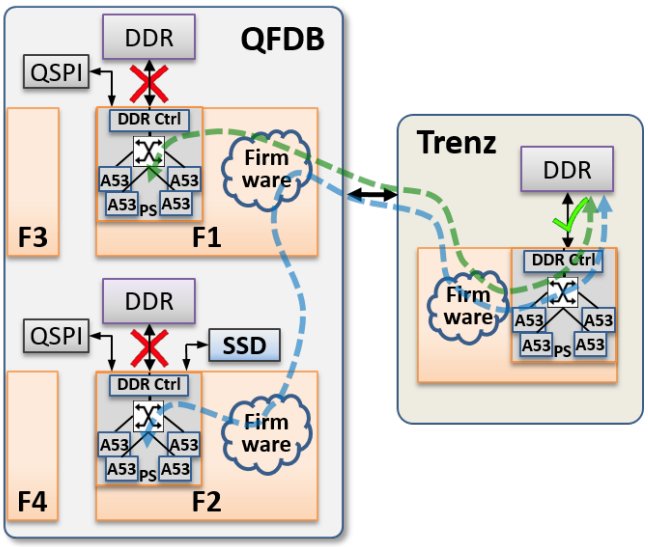


Fig. 4: RAM-less OS boot through memory borrowing

squeezed into the 32 MB of the QSPI flash memory. To that end, the U-Boot loader was trimmed aggressively, a custom Linux kernel (v4.9) configuration was used, and a minimal BusyBox setup was instantiated in the form of an initramfs image. The overall storage space used per node reached a mere 20 MB, including a 16 MB compressed bitstream. The First Stage Boot Loader (FSBL) was also modified not to require external RAM, and the code re-location of the U-Boot loader was bypassed.

On the firmware side, complementary designs were built on the QFDB and the Trenz board. Following Figure 4, any memory access from the QFDB MPSoC is steered to the Programmable Logic (PL), instead of reaching the local DDR-RAM. The memory access then exits the FPGA through a GTH transceiver. When reaching the other end, the access is transformed using a specific IP, and reaches the functioning remote DDR through the Processing System (PS). The response then travels back in a similar manner. The whole mechanism is transparent to the software execution environment. The transceivers operate at their maximum lane rate (10.3125 Gb/s), which offers a reasonable memory throughput for moderate workloads.

Since only the Network FPGA of the QFDB is directly connected to the outside world, as seen in Figure 1, additional effort was required for the boot of the other FPGAs. Hence, the bitstream of the Network FPGA is enhanced to also acts as a proxy, which forwards transactions for the other three nodes, and steers them to discrete memory ranges on the other end, as shown in Figure 4. With this approach, the F2 "Storage" FPGA was successfully booted, with the Linux accessing its physical memory through the F1 intermediate hop.

V. Multi-board prototype integration

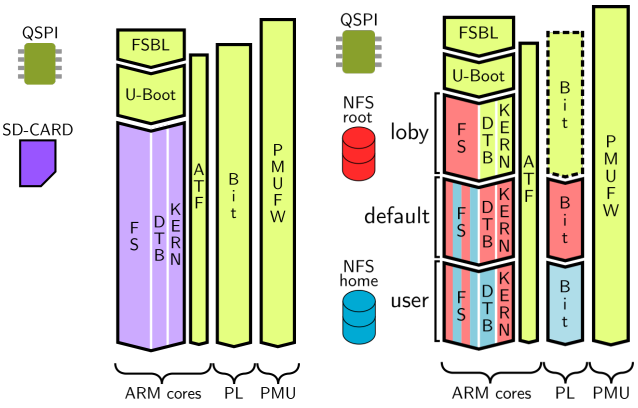
A prototype is being built based on the QFDB. In Table II, indicative numbers of the dimension of this prototype are given, highlighting the scale of this effort. Under current estimates, the final prototype will be composed of a thousand cores, 4 TB of DDR4-RAM, and 16 TB of SSD storage.

A. Software environment

The utilization of a prototype consisting of tens of complex HPC multi-node boards is usually tedious, in particular when different firmware and system software versions are in use. In this section, we provide insights into the software solutions that we developed to significantly ease up the life of users. The boot process has been customized to support versatile research activities and fast deployment of boot images. Tools have also been developed to easily generate boot images and boot packages. Finally, a Linux distribution was developed for the needs of the platform.

TABLE II: Prototype dimensions

Prototype	Boards	Zynqs	Cores	RAM [GB]	NVM [TB]
1 QFDB	1	4	16	64	0.25
Current	48	192	768	3072	12
Final (est.)	64	256	1024	4096	16



(a) Typical (b) Our system

Fig. 5: Zynq Ultrascale+ Boot sequence comparison

In Figure 5, a comparison between a typical boot sequence and the one used by the Zynqs of the QFDB is presented. In both cases, the First Stage Boot Loader (FSBL), the PMU firmware (PMUFW), the ARM Trusted Firmware (ATF) and BL33 (U-Boot) are all retrieved from the local QSPI flash memory. In the typical boot sequence, the PL bitstream (Bit) is also loaded at this point. The kernel (KERN), Device Tree (DTB) and file system (FS), which are necessary for Linux operation, are read from an SD-Card. In our system, the BL33 is stripped to a strict minimum, and a trimmed Linux kernel nicknamed *loby* is used to boot the full Linux system, thus providing better system flexibility. The *loby* kernel mounts a root NFS storage common to all boards of the prototype. This rootfs contains a configuration file that maps each QFDB to a board class, based on the unique Board ID presented in Section IV-B. Once the board class and the node ID within the QFDB have been retrieved, the corresponding boot package is loaded. Each boot package usually contains a bitstream and a device tree, and it might also include a different kernel and initramfs. The selected configuration is then loaded by using the *kexec* command [32]. At a later time, the user has the possibility to apply its own boot package to the node, without rebooting the board. The rootfs is usually mounted as read-only by the kernel, and a *tmpfs*-based read/write overlay of the filesystem is mounted, allowing multiple users to transparently share the same remote filesystem without risking mishaps in the prototype environment.

We also developed a tool called *yat* (yet another tool) to readily generate boot images using different patch profiles. After the patches have been applied, custom cross-compiling toolchains (generated with *crosstool-ng*) are used for building the binaries. *yat* is used to create both the image file to be flashed to the SPI-NOR flash attached to each FPGA, and the boot package file described earlier.

For our prototype, a Linux distribution called Carvoonix was created, based on Gentoo Linux. In effect, HPC nodes alike the QFDB are unique environments in many ways. The software requirements are very different from desktops, servers and embedded devices. In addition, our prototype is not only a

production environment but also a development environment, both for us and our partners. Binary Linux distributions are not built with this requirement in mind, and make the maintenance of such a system very complex.

Our distribution combines the flexibility of Gentoo by having a builder version used for carefully create selected binary packages from source into a binary repository, and a normal version that fetches packages from the binary repository.

B. Multi-board Communication Infrastructure

In order to support versatile research activities on our prototype, a flexible infrastructure presented in Figure 6 was developed. On one hand, a Manager PC is used to manage the boards and provide various services. On the other hand, additional software and firmware were designed to bring Ethernet access to each of the Zynqs of the QFDB. Finally, boards can be connected amongst themselves by means of an ad-hoc interconnect.

In order to ease deployment of software and firmware on the boards, a Manager PC carries out diverse functionalities. Foremost, the PC provides DNS and DHCP services to the boards, through the *dnsmasq* program. Second, it provides the NFS shared folders needed to boot the QFDBs, as discussed in Section V-A. Third, the Manager PC acts as a SSH gateway, which lets users connect to the boards easily.

On the QFDB side, a substantial effort was made to cope with various use cases, as illustrated in Figure 6. In QFDB 1, the bitstream of F1 includes a custom 10 Gigabit Ethernet (10GbE) MAC IP. A minimal Ethernet switch routes Ethernet packets to destination FPGA(s). Each FPGA uses DMA engines to transfer Ethernet packets to and from the Processing System (PS). Alternatively, as in the case of QFDB 3, a software bridge in F1 may provide Ethernet access to the other FPGAs, through the PS Gigabit Ethernet Module (GEM). A custom tool was written to configure the F1 network because standard tools such as *brctl* or *ip* would cause disruption of the access to the NFS-based rootfs. Finally, as seen in QFDB 2, F1 may be connected to the LAN through the 1GbE interface while F2−F4 utilize the 10GbE switch described earlier. In each case, the MAC address of every interface is seeded with the eFUSEs scheme discussed in Section IV-B. However, since access to the eFUSEs is only possible from the secure world, an OEM service was added to the ATF Secure Monitor, and a kernel driver issues Secure Monitor Calls (SMCs) to retrieve the information and expose it through *procfs* to the user space. As a consequence, the Ethernet network environment is set very early in the kernel boot, which is necessary to mount the NFS-based rootfs.

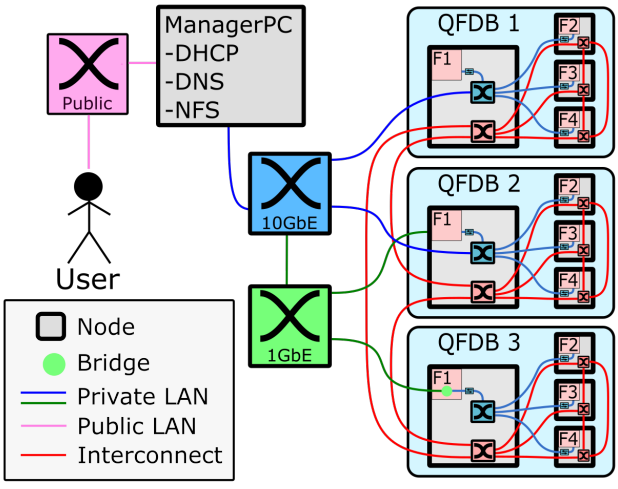


Fig. 6: Network infrastructure of our prototype

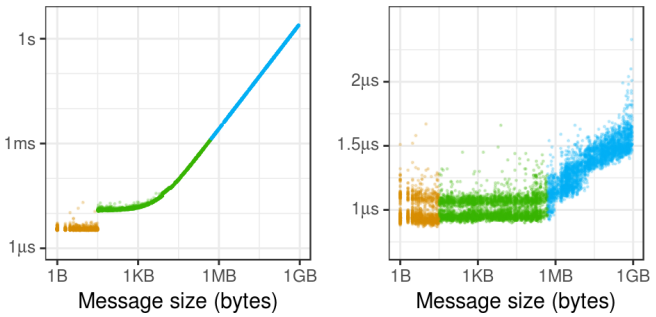
VI. Impact of the QFDB

In this section, we highlight the impact of the QFDB on a number of ongoing research activities. In effect, the QFDB has been used extensively for the last two years, and a variety of achievements were obtained by leveraging its capabilities. In the ExaNeSt project [1], it was leveraged to develop novel HPC interconnects, and validate a high-density liquid-cooled HPC rack . In EcoScale, the QFDBs were utilized to implement an innovative scheme that allows hardware acceleration across distributed fabric of interconnected FPGAs. Finally, various hardware acceleration efforts took place on QFDBs.

A. Advanced Interconnect

The QFDB is a first-class substrate to develop cutting edge interconnects. In effect, the board presents 10 links to the outside world, that is enough for most common topologies. The Programmable Logic of the Zynqs can be used to implement both Network Adapter functionalities (e.g., RDMAs, mailboxes) as well as ad-hoc switches and links. The tightly coupled Processing System can either be used as a network node or to implement Control Plane functionalities, with much better agility and performance than when a PCIe bus is involved. In addition, the density of the board and the software environment presented in Section V-A allow for the quick deployment of significant interconnect topologies.

In the ExaNeSt project [1], the ExaNet interconnect [7] was developed based on the QFDBs features. The low-latency 3D-torus high-speed fabric was brought up and optimized by leveraging the environment described in Section V. The drivers and an MPI library were also developed in this context,



(a) Ping-pong completion (b) Non-blocking send execution

Fig. 7: ExaNet MPI stack performance

and large scientific applications exploited this stack. Figure 7 shows the performance of the current version of the ExaNet stack between the Network FPGA of two adjacent QFDBs. With the current version of the firmware and software stack, the one-way latency observed by the application for small messages is one microsecond. Figure 7a shows the completion time for an MPI ping-pong with different message sizes. For small packets, the round trip lasts on average 3 μs, by leveraging custom packetizer and mailbox IPs. For larger messages, the custom DMA engine sustains high-throughput while keeping a modest initialization overhead. In Figure 7b, the execution time of a non-blocking send (*MPI_ISend*) is shown for different message sizes in the ExaNet implementation. It remains stable around 1 μs for transfers up to 1 GB, while in many vendor implementations, it tends to increase substantially even for much smaller messages. With respect to the capabilities of the Zynq Ultrascale+, this level of performance is very encouraging, and provides a very positive insight regarding the ExaNet potential.

At present, QFDBs are utilized to pursue complementary research activities in the field of HPC interconnects such as congestion control, multi-path routing, alternative topologies evaluation, and network functions acceleration.

B. Dense HPC system

One of the important challenges for future HPC systems is to reach sufficient computational and power density to keep the dimensions of machines reasonable. Within the ExaNeSt project, a high-density liquid-cooled rack was assembled. As a consequence, the QFDB specifications were driven by strong density constraints. The rack, including its cooling technology, power delivery and high-speed signal conditioning, were designed by Iceotope [34]. The proprietary cooling technology allows for clean removal and insertion of liquid-cooled *blades*. The compute nodes are sealed within the *blade* enclosure. Figure 8 shows a picture of the ExaNeSt testbed. Each *blade* hosts 4 QFDBs. The Ethernet infrastructure described in Section V-B is implemented by means of commercial switches (along with firmware and software elements in the QFDBs). The cabling implements the ExaNet interconnect discussed in Section VI-A.

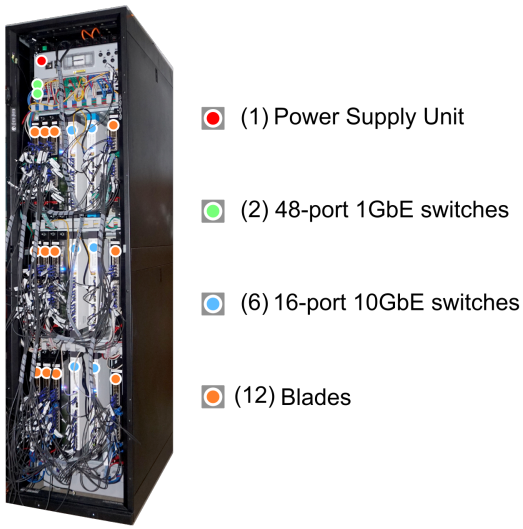


Fig. 8: The ExaNeSt HPC testbed

C. Reconfigurable acceleration on shared multi-FPGA resources

The QFDBs were also deployed for the prototyping of the UNILOGIC architecture, introduced in the ECOSCALE H2020 project. This approach supports efficient sharing of distributed reconfigurable logic across the system [8], [9], [33]. In order to implement efficient sharing, the architecture supports partitioned global address space so that a) the hardware accelerators on the FPGAs of the QFDB boards can be accessed directly by any processor in the system, and b) the hardware accelerators can access any memory in the system. In this way, the architecture offers a unified environment where all the system resources can be seamlessly accessed by the Operating System of any processor. The architecture is tailored to the characteristics of an HPC environment, partitioning the design into several nodes that communicate through a fat-tree infrastructure, as depicted in Figure 9. Each node is an entire sub-system including a processing unit, memory, storage and reconfigurable resources that can be accessed by any node in the system.

The reconfigurable resources are split into a static partition, which provides the communication infrastructure, and four fixed-size slots that can be reconfigured and accessed independently or combined together (slot merging), in order to support fine or coarse grain partitioning and utilization of the reconfigurable resources. Partial runtime reconfiguration has been employed to dynamically reconfigure the accelerator slots. These slots can be remotely reconfigured and accessed directly by any processor in the system using the Xilinx Internal Configuration Access Port (ICAP) that resides in the UNILOGIC global address space. The tight coupling of the resources on the QFDB allows an accelerator on any reconfigurable slot to access any DDR memory in the system with minimal communication overhead. A runtime system monitors the system status and manages the reconfigurable resources across the whole platform.

The QFDB has provided an ideal development platform offering reconfigurable resources that are tightly coupled

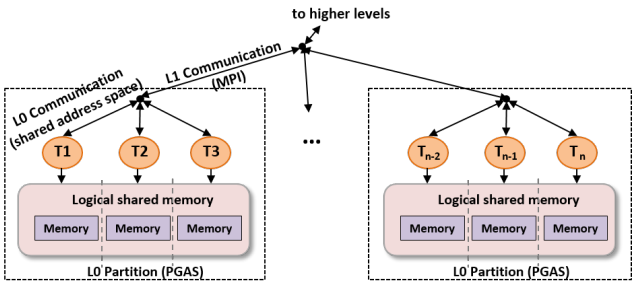


Fig. 9: QFDBs in a fat-tree infrastructure

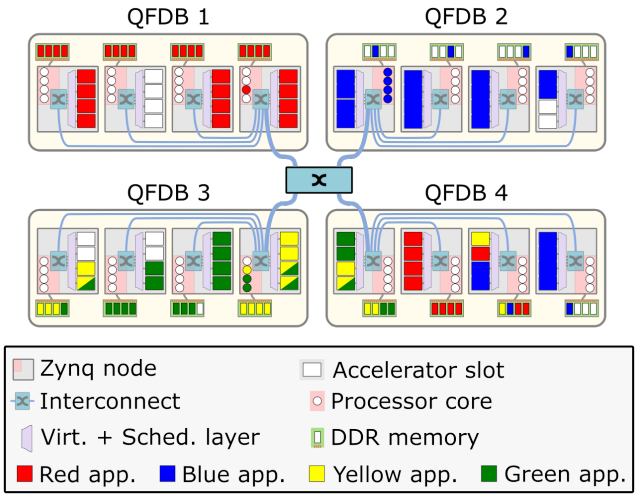


Fig. 10: Example UNILOGIC infrastructure

through low-latency, high-speed connectivity. Each QFDB FPGA represents a node of the UNILOGIC architecture while the communication infrastructure has been implemented by interconnecting 16 QFDB boards together. In order to connect multiple QFDB boards together, the ECOSCALE project designed and implemented a carrier board that can support up to eight interconnected QFDB boards, and thus up to 32 FPGAs, within an 1U chassis. The architecture can easily be extended by further interconnecting multiple chassis.

Figure 10 presents a small UNILOGIC system with four interconnected QFDBs. In each Zynq Ultrascale+, the UNILOGIC firmware also implements four dynamically reconfigurable accelerator slots, that are accessible to applications through a virtualization and scheduling layer. In this example, resources are shared among four applications. The software part of the Red application runs on a node of QFDB 1 (red circle in the rightmost FPGA), and spreads its data in the memory of five other nodes (red memory DIMMs on top). Accelerators for this application (red slots inside FPGAs) are spawned close to the data, however all accelerators can also access remote memory with a moderate latency. Another application, the Blue one, requires two coarse grain accelerators that require respectively the merging of 2 and 4 slots. Two more applications, the Green and Yellow, share QFDB 3. Since the UNILOGIC virtualization and scheduling layer allows for seamless sharing of accelerators across applications, they also reuse a common accelerator function (double-colored slots). Finally, all applications use resources of QFDB 4, illustrating the flexibility offered by the UNILOGIC architecture.

D. Stand-alone accelerators

The reconfigurable resources in a QFDB's FPGAs and their AXI memory interfaces with the Processing System (PS) enable the deployment of tightly coupled accelerators. To showcase the capabilities of our prototype, High-Level Synthesis (HLS) was used to develop a Matrix Multiplication accelerator, as it is a very common kernel in HPC and AI applications – for instance GEMM in BLAS Level-3 routines and also in Convolutional Neural Networks.

Xilinx's Vivado HLS was used to develop a highly optimized matrix multiplication kernel tile using the directive-oriented style of HLS. This kernel tile is parameterizable with different sizes, in order to experiment with implementations that have different area *vs.* performance trade-offs. The kernel tile operates on single-precision floating point elements (FP32) and is tuned considering the finite reconfigurable resources (LUTs, Flip-Flops, DSPs) and the internal memory (BRAM) size limitations. Moreover, its design used efficiently the provided AXI memory bandwidth.

The basic matrix multiplication algorithm is shown below:

```
for i=0 to n do
  for j=0 to n do
    for k=0 to n do
      C[i][j] += A[i][k] x B[k][j]
```

Given the finite resources of the FPGA and the need to accelerate matrix multiplication of very large arrays that do not fit in internal memory (BRAM), a tiled approach was followed and the optimized kernel was applied over the corresponding tiles of the original arrays. Based on the study of the associated area-speed trade-offs, a tile size of 128×128 operating at 300 MHz was selected. The kernel tile implementation unrolls the k loop completely, performing 128 FP32 multiplications and 128 FP32 additions per cycle (fully-pipelined), which is similar to a vector unit operating on 4096 bits. Moreover, the j loop is partially unrolled by a factor of 4, which is similar to having 4 vector units of 4096 bits. In total, the kernel tile performs 512 FP32 multiplications and 512 FP32 additions per cycle (fully-pipelined). Once the appropriate elements of the arrays are stored in on-chip accelerator memory (BRAMs), the tile execution lasts $\sim$4200 clock cycles. Three AXI HP ports (one per array) are utilized to load data into the local BRAMs, in order to continuously feed the kernel with data and match the required bandwidth. By performing a series of performance optimizations on the memory interfaces and load/unload techniques, each FPGA managed to achieve *275 FP32 GFLOP/s* and offload a matrix multiplication application from Linux. With all four FPGAs, a single QFDB can perform and sustain *more than 1 FP32 TFLOP/s*.

The matrix multiplication FP32 kernel tile of 128×128 operates at 300 MHz, and requires 153K LUTs, 300K Flip-Flops, 2057 DSPs and 416 BRAMs. The resource utilization of each FPGA device implementing our kernel tile is 56% of LUTs, 55% of Flip-Flops, 46% of BRAMs and 82% of DSPs. Moreover, power consumption of the accelerator was measured using the plethora of power sensors residing on the QFDB. Based on the readings of these sensors, encouraging power efficiency results were gathered. These measurements show that the dynamic power consumption of the accelerator is 16.2 Watts, which yields 17 FP32 GFLOP/s per Watt.

VII. Conclusions

In this paper, we presented the Quad-FPGA Daughter Board (QFDB) architecture and the system software and firmware set up to support our research activities running on multiple interconnected boards in a large-scale system. The design and the bring-up of the QFDB was very challenging and we discussed aspects essential for the development of complex System-On-Modules. Finally, we described several achievements that exploited the QFDB, and highlight its potential.

In effect, the QFDB takes advantage of the reconfigurable logic and connectivity resources included in modern MPSoCs to design a unique high-density module, that can serve as a building block for prototyping large-scale systems. In addition, the QFDB is powerful enough to accelerate compute-intensive workloads. For example, large scientific application runs were achieved [7], based on a custom interconnect and MPI stack. The density of the module was also leveraged to build a high-density liquid-cooled rack presented in Section VI-B. Moreover, the Programmable Logic of the Zynq devices was used to run the accelerator presented in Section VI-D, that could execute up to 1.1 FP32 TFLOP/s on a single QFDB. Finally, multiple QFDBs were utilized to develop the UNILOGIC architecture, that allows for the seamless sharing of Programmable Logic and memory resources across multiple compute nodes [8].

The flexibility and density of the QFDB paves the way to large-scale system prototyping and hardware acceleration, suitable for a large spectrum of research activities, from hardware acceleration to high-performance interconnects, and from runtime software to accelerated HPC applications.

VIII. Acknowledgements

This work was carried out within the ExaNeSt (FET-HPC) project, funded by the European Union's Horizon 2020 research and innovation programme under grant agreement № 671553, and the presentation of the work was funded by the Eurolab4HPC project under grant agreement № 800962.

References

[1] M. Katevenis, N. Chrysos, M. Marazakis, I. Mavroidis, F. Chaix, N. Kallimanis, J. Navaridas, J. Goodacre, P. Vicini, A. Biagioni, P. Stanislao Paolucci, A. Lonardo, E. Pastorelli, F. Lo Cicero, R. Ammendola, P. Hopton, P. Coates, G. Taffoni, S. Cozzini, M. L. Kersten, Y. Zhang, J. Sahuquillo, S. Lechago, C. Pinto, B. Lietzow, D. Everett and G. Perna, "The ExaNeSt Project: Interconnects, Storage, and Packaging for Exascale Systems", 2016 Euromicro Conference on Digital System Design, DSD 2016, Limassol, Cyprus, August 31 - September 2, 2016.

[2] K. Oner, L. A. Barroso, S. Iman, J. Jeong, K. Ramamurthy and M. Dubois, "The Design of RPM: An FPGA-based Multiprocessor Emulator", In Proceedings of the 3rd ACM International Symposium on Field-Programmable Gate Arrays, 1995.

[3] C. Chang, J. Wawrzynek, R. W. Brodersen, "BEE2: A High-End Reconfigurable Computing System", IEEE Design and Test of Computers, vol. 22, no. 2, 2005.

[4] J. Wawrzynek, M. Oskin, C. Kozyrakis, D. Chiou, D. A. Patterson, S. Lu, J. C. Hoe and K. Asanovi, "RAMP: A Research Accelerator for Multiple Processors", Technical Report UCB/EECS-2006-158, EECS Department, University of California, Berkeley, Nov. 2006.

[5] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman and J. Luo, "NetFPGA - An Open Platform for Gigabit-rateNetwork Switching and Routing", In Proceedings of the IEEE International Conference on Microelectronic Systems Education, 2007.

[6] R. F. Barrett, S. Borkar, S. S. Dosanjh, S. D. Hammond, M. Heroux, X. S. Hu, J. Luitjens, S. G. Parker, J. Shalf and L. Tang, "On the role of co-design in high performance computing", Advances in Parallel Computing, 2013.

[7] M. Ploumidis, N.D. Kallimanis, M. Asiminakis, N. Chrysos, V. Papaeustathiou, P. Xirouchakis, M. Gianoudis, N. Dimou, A. Psistakis, P. Peristerakis, G. Kalokairinos and M. Katevenis, "Software and Hardware co-design for low-power HPC platforms", ExaComm 2019.

[8] I. Mavroidis, I. Papaefstathiou, L. Lavagno, D. S. Nikolopoulos, D. Koch, J. Goodacre, I. Sourdis, V. Papaefstathiou, M. Coppola and M. Palomino, "ECOSCALE: Reconfigurable computing and runtime system for future exascale systems", DATE 2016.

[9] A. Ioannou, P. Malakonakis, K. Georgopoulos, I. Papaefstathiou, A. Dollas and I. Mavroidis, "Optimized FPGA Implementation of a Compute-Intensive Oil Reservoir Simulation Algorithm", SAMOS 2019.

[10] The EuroEXA Project, https://euroexa.eu, 2017-2020.

[11] A. Rigo et al, "Paving the Way Towards a Highly Energy-Efficient and Highly Integrated Compute Node for the Exascale Revolution: The ExaNoDe Approach", DSD 2017.

[12] M. Marazakis, J. Goodacre, D. Fuin, P. Carpenter, J. Thomson, E. Matus, A. Bruno, P. Stenstrom, J. Martin, Y. Durand, and Y. Dor, "EUROSERVER: Share-Anything Scale-Out Micro-Server Design", DATE 2016.

[13] "The Convey HC-1: The World's First Hybrid-Core Computer" (2009), http://xilinx.eetrend.com/files-eetrend-xilinx/news/200912/238-214-hc-1.pdf

[14] R. Huizen, "FPGAs in the Cloud: Should you Rent or Buy FPGAs for Development and Deployment?" White Paper, BittWare, a Molex Company, www.bittware.com/resources/fpgas-in-the-cloud/

[15] https://www.hitechglobal.com/

[16] https://www.digilent.com

[17] 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", HPEC 2016.

[18] https://aws.amazon.com/ec2/instance-types/f1/

[19] O. Pell and O. Mencer, "Surviving the End of Frequency Scaling with Reconfigurable Dataflow Computing", SIGARCH Comput. Archit.News, vol. 39, no. 4, Dec. 2011.

[20] www.maxeler.com/technology/dataflow-computing/

[21] https://www.sciengines.com/

[22] C. Selvidge and V. Chobisa, "The Veloce Strato Platform: Unique core components create high-value advantage", Mentor, a Siemens business, www.mentor.com/products/fv/emulation-systems/

[23] "Cadence Palladium Z1 Enterprise Emulation Platform", www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/palladium-z1-ds.pdf

[24] J. Weerasinghe et al., "Network-attached FPGAs for data center applications", FPT 2016.

[25] "BEEcube FPGA Based Rapid Prototyping Platforms", http://docplayer.net/33010877-Beecube-fpga-based-rapid-prototyping-platforms-for-military-communications.html

[26] "quad SG-280 Prototyping System" product brief, www.profpga.com/files/profpga_quadsg280_product_brief_3.pdf

[27] "Datasheet of the Samsung 960 PRO M.2", www.samsung.com/semiconductor/global.semi.static/Samsung_SSD_960_PRO_Data_Sheet_Rev_1_2.pdf.

[28] J. Dongarra and P. Luszczek, "HPC Challenge: Design, History, and Implementation Highlights", book chapter in 'Contemporary High Performance Computing: From Petascale Toward Exascale' edited by J. S. Vetter, Taylor & Francis group, 2013, isbn 978-1-4665-6834-1.

[29] https://www.cs.virginia.edu/stream/

[30] https://git.kernel.org/pub/scm/linux/kernel/git/axboe/fio.git

[31] "Technical Reference Manual of Trenz TEBF0808", https://wiki.trenz-electronic.de/display/PD/TEBF0808+TRM.

[32] "Kexec", https://en.wikipedia.org/wiki/Kexec.

[33] P. Malakonakis, K. Georgopoulos, A. Ioannou, L. Lavagno, I. Papaefstathiou and I. Mavroidis, "HLS Algorithmic Explorations for HPC Execution on Reconfigurable Hardware - ECOSCALE", ARC 2018.

[34] "Iceotope website", https://www.iceotope.com.