

The TEXTAROSSA Project: Cool all the Way Down to the Hardware

Antonio Filgueras^{*†}, Giovanni Agosta^{‡xii}, Marco Aldinucci^{xiii xvi}, Carlos Álvarez^{*†}, Pasqua D’Ambra^{xiii},
Massimo Bernaschi^{.xiii}, Andrea Biagioni[§], Daniele Cattaneo[‡], Alessandro Celestini^{.xiii}, Massimo Celino^{||},
Carlotta Chiarini^{§xiv}, Francesca Lo Cicero[§], Paolo Cretaro[§], William Fornaciari^{‡xii}, Ottorino Frezza[§],
Andrea Galimberti[‡], Francesco Giacomini^{xv}, Juan Miguel de Haro Ruiz^{*†}, Francesco Iannone^{||}, Daniel Jaschke^{x ‡‡‡††},
Daniel Jiménez-González^{*†}, Michal Kulczewski[¶], Alberto Leva[‡], Alessandro Lonardo[§], Michele Martinelli[§],
Xavier Martorell^{*†}, Simone Montangero^{††xi ‡‡x}, Lucas Morais^{*†}, Ariel Oleksiak[¶], Paolo Palazzari^{||}, Luca Pontisso[§],
Federico Reghenzani^{‡xii}, Cristian Rossi^{§xiv}, Sergio Saponara^{**‡xii}, Carlo Saverio Lodi[‡], Francesco Simula[§],
Federico Terraneo^{‡xii}, Piero Vicini[§], Miquel Vidal[‡], Davide Zoni[‡], Giuseppe Zummo^{||}

^{*}Universitat Politècnica de Catalunya, Spain [name].[surname]@upc.edu

[†]Barcelona Supercomputing Center, Spain [name].[surname]@bsc.es

[‡]DEIB – Politecnico di Milano, Italy [name].[surname]@polimi.it

[§]INFN, Sezione di Roma, Italy [name].[surname]@roma1.infn.it

[¶]Poznan Supercomputing and Networking Center, Poland [name].[surname]@man.poznan.pl

^{||}ENEA, Italy [name].[surname]@enea.it

^{**}University of Pisa, Italy [name].[surname]@unipi.it

^{††}Dipartimento di Fisica e Astronomia "G. Galilei", Università degli Studi di Padova, Italy

^{‡‡}INFN, Sezione di Padova, Italy [name].[surname]@pd.infn.it

^xInstitute for Complex Quantum Systems, Ulm University, Germany

^{xi}Padua Quantum Technologies Research Center, Italy

^{xii}Centro Interuniversitario Nazionale per l’Informatica (CINI), Italy

^{xiii}National Research Council (CNR), Italy

^{xiv}Università "Sapienza" di Roma, Italy

^{xv}INFN, CNAF, Italy

^{xvi}University of Torino, Italy [name].[surname]@unito.it

Abstract—The TEXTAROSSA project aims to bridge the technology gaps that exascale computing systems will face in the near future in order to overcome their performance and energy efficiency challenges. This project provides solutions for improved energy efficiency and thermal control, seamless integration of heterogeneous accelerators in HPC multi-node platforms, and new arithmetic methods. Challenges are tackled through a co-design approach to heterogeneous HPC solutions, supported by the integration and extension of HW and SW IPs, programming models, and tools derived from European research.

Index Terms—High-performance computing, heterogeneous computing, FPGA, GPU, thermal management, power management, hardware accelerators, programming models

I. INTRODUCTION

High-Performance Computing (HPC) technologies are key to support several applications in domains such as computational fluid dynamics, weather forecasting, bioinformatics and Artificial Intelligence (AI). With the recent explosion in HPC for Artificial Intelligence (HPC-AI), the trend in the design of HPC infrastructures is leaning more and more toward heterogeneous HW architectures in response to larger performance demands as well as the need to improve energy efficiency to achieve "Green HPC". We address the challenge

of increased performance while remaining within power and energy bounds with a holistic approach taking into account multiple factors across the HPC HW/SW stack. This includes analysis and redesign of applications to use more efficient reconfigurable application-specific accelerators, development of such accelerators, management of resources, design of the underlying infrastructure and cooling and management of such infrastructure. By approaching the whole HW/SW stack, higher-level SW components can affect how HW and infrastructure are designed while infrastructure can also affect application design to reach energy efficiency and performance targets. TEXTAROSSA is a three-year project co-funded by the European High Performance Computing (EuroHPC) JU.¹ The project is led by ENEA (Italy) and aggregates 17 European partners.² The paper is organized as follows. Section II describes the HW platforms designed during the project. Section III describe project contributions to the HW/SW stack. Section IV presents evaluation of the contributions across different use cases. Finally, section V draws some conclusions.

¹<https://eurohpc-ju.europa.eu/>

²<https://www.textarossa.eu>

II. HARDWARE PLATFORMS

During the project, we developed two different experimental HW platforms known as Integrated Development Vehicles (IDVs). Both prototypes, named IDV-A and IDV-E, incorporate commercially available components, the descriptions of which are provided in sections II-A and II-B, respectively. All of these platforms utilize the project-designed two-phase cooling technology described in Section III-A.

A. IDV-A

The IDV-A prototype is based on Atos Sequana3 platform. It consists of one Nvidia Redstone-Next GPU board equipped with four Nvidia H100 GPUs. These GPUs are attached to the host system using PCIe 4.0 x16. Moreover, each GPU is connected with all other GPUs using 4x NVLink providing GPU-to-GPU transfers up to 200 GB/s between each pair of GPUs. The motherboard is an Atos C4E CPU board equipped with two Intel Xeon 8470 CPUs. The total Thermal Design Power (TDP) dissipated by a single node can reach more than 3500 W. Each of the CPUs dissipates up to 350 W while each of the H100 GPUs can dissipate up to 700 W.

B. IDV-E

The IDV-E prototype, developed by E4, is based on the Ampere Mt.Collins 2u system. It is equipped with two Ampere Altra Max ARMv8 processors and two AMD Alveo U280 FPGA accelerator cards that are attached via PCIe 4.0 x8 or PCIe 3.0 x16. Both FPGAs are connected via two QSFP+ links that are capable of providing up to 100 Gb/s in full duplex mode between FPGAs each. Each of the CPUs can dissipate up to 250 W while each of the FPGA cards can dissipate up to 225 W for a total of 950 W per node.

III. PROJECT CONTRIBUTIONS

A. Evaporative cooling and thermal management

Availability, cost, and performance of current HPC platforms are constrained by thermal considerations, necessitating optimized heat dissipation solutions with runtime thermal modeling and control policies for reliable and efficient operation [1]. In terms of *heat dissipation improvements*, InQuattro has developed and patented an innovative thermal management solution based on two-phase mechanically pumped loops. This solution utilizes flow boiling heat transfer to cool electronics more efficiently. By harnessing the latent heat of vaporization, this approach significantly reduces flow rates, maintains small temperature gradients, and increases heat transfer coefficients compared to both air cooling and traditional liquid cooling systems. Two-phase cooling systems using evaporation and condensation are recognized as the best way to meet demanding cooling requirements in terms of compactness, weight, and energy consumption [2]. One of the main achievements of TEXTAROSSA with IDV-A and IDV-E has been demonstrating the feasibility of integrating server-level two-phase cooling solutions for heterogeneous computing. The installation on IDV-E showcased how it is possible to seamlessly integrate the evaporative cooling on top of existing platforms. Similarly,

on IDV-A, it was demonstrated that a hierarchical thermal control approach is feasible, where control over the Dynamic Voltage and Frequency Scaling (DVFS) of the CPU effectively cooperates with outer control of the evaporative cooling to achieve efficient global thermal management. Fig. 1 depicts the installation on the IDV-A Sequana3 server platform as described in Section II.

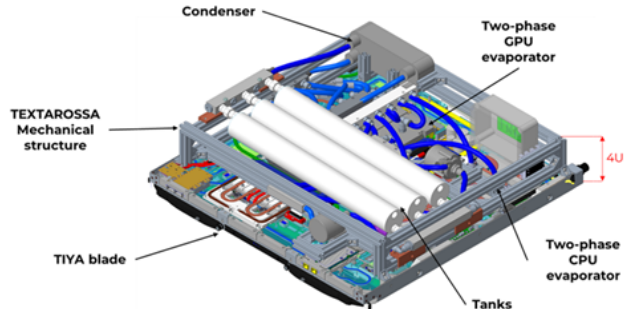


Fig. 1: Evaporative Cooling installed on IDV-A.

Concerning the *thermal modeling*, which is the cornerstone of any control policy, POLIMI developed a transient model using Equation-Based Object-Oriented Modeling (EB-OOM) technology, and capable of performing co-simulation with the 3D-ICE chip thermal simulation [3]. The goal of a thermal model is the accurate reproduction of the chip temperature profile when subject to a given power dissipation stimulus, and under prescribed boundary conditions, which include the heat dissipation solution surrounding the chip as well as its connection to the ambient. When modeling a closed cooling cycle, the boundary conditions include all the parts that compose the cycle, thus pipes, tanks, the condenser/heat exchanger, and pumps. However, these components can be modeled with a coarser level of detail, preferring lumped modeling rather than detailed models including spatial discretization. The chip model is connected to the EB-OOM model of the evaporator and coolant cycle with a fixed coolant flow rate. The spatial discretization of the chip is $320 \times 320 \times 2$ finite volumes, the evaporator base plate is 12×12 finite volumes, and the coolant flow within the evaporator is discretized in 9 finite volumes. This level of resolution allows to create fine grain thermal maps of the surface of the chip, enabling fine-grain control over the parameters steering the maintaining of the status of phase change of the coolant, that is not a trivial task: optimal performance occurs when both the flow rate and the dissipated power are either high or low simultaneously.

A *hierarchical thermal controller* has been developed by POLIMI to limit the operating temperatures of computational devices while taking advantage of the evaporative cooling technology to also limit the performance degradation due to frequency reduction. For massively multicore architectures CPUs, such as the Intel Xeon Platinum in IDV-A with 52 cores, each core is equipped with one temperature sensor and one frequency actuator. This setup enables fine-grained thermal control at the core level. Differently, Nvidia H100

GPUs only provide a single temperature sensor and frequency actuator per GPU. Therefore, in this case, thermal control is implemented at the level of the entire GPU. Additionally, the thermal control strategy needs to take into account the presence of an evaporative cooling system. It has additional actuators (pumps) to set the evaporative coolant liquid flow rate and additional sensors to monitor coolant temperatures and measure flow rate. The flow rate can differ from the prescribed one due to head losses in the system that largely depend on the coolant vapor quality. Due to the involved thermal inertia, the evaporative cooling system is considerably slower compared to the on-chip phenomena. For the above-mentioned reasons, namely, the presence of multiple sensors and actuators operating at different timescales, a single global control loop is an unfeasible option to solve the thermal control problem. The implemented solution thus hierarchically splits the control problem into one fast inner control loop per controlled device (CPU core/GPU), plus one evaporative controller per cooling cycle, each having an additional inner control loop to actuate the pump while compensating for variable head losses. The fast inner control loop design is based on an event-based thermal control policy that is patented by POLIMI [4].

For the CPU we used Model-Specific Registers (MSRs) for sensing temperature and computational load, and to drive the DVFS actuator. For what concerns controlling the operations of the GPUs, we exploited the Nvidia Management Library (NVML). Interfacing with the flow meter sensors and pump actuators was instead performed through custom code written in cooperation with InQuattro. The inner control loops are placed just above the HW driver layer. For the CPUs and GPUs, we developed a C++ implementation of the event-based controller. A different number of those controllers can be instantiated to accommodate the number of CPUs and GPUs present in the computing architecture. The CPU and GPU controllers have been tuned separately based on dedicated identification experiments, due to the different thermal dynamics between the two types of HW. The controller, validated under stress conditions, has been capable of maintaining the GPU temperature within 1°C of the target set point, and for all 104 CPU cores within less than 2°C of the target set point.

B. Hardware IPs for task scheduling

One source of inefficiencies in heterogeneous systems is task management. The overhead of keeping track of task data dependencies and sending tasks to accelerators in some cases can negate all improvements achieved by using application accelerators. To solve this issue, BSC in collaboration with UPC developed a HW Intellectual Property core (IP) that takes care of scheduling tasks into different cores or accelerators. This serves two purposes. On one hand, management jobs such as keeping track of the accelerator state are offloaded to an external IP, freeing host resources. On the other hand, it reduces the amount of communication and synchronization points between the different processing elements in a given system. One of the use cases for this IP is to manage execution

in multiple FPGA accelerators. In this case, the scheduler IP is placed in the FPGA, close to the accelerators. This allows for fast and efficient management of accelerators, furthermore, it also allows accelerators to spawn tasks to be executed in other accelerators without host intervention, allowing efficient management of a large number of fine-grained tasks as most host-device synchronization is removed. In this scenario, the host does not interact directly with accelerators, but with the task scheduler, which manages the accelerators.

We also have integrated this IP in a Rocket Chip-based RISC-V multicore system [5]. In this case, the HW task scheduler is tightly coupled with the cores. More precisely a new CPU instruction is added so that CPU cores can access task scheduling HW allowing efficient task scheduling for multicore systems. In this case, all cores access the HW scheduler through an arbitration module. This largely removes the need to use synchronization primitives between cores. Tasks are scheduled based on their data dependencies automatically by the HW module.

C. OmpSs@FPGA multi-node task-based programming model

We improved HW implementation quality of results as well as memory access efficiency in the OmpSs@FPGA programming model. This results in an overall improvement in performance and energy efficiency [6]. We also added support to multiple FPGA devices, which implies managing groups of accelerators attached to different memory spaces (i.e. each board memory) for a single node. On top of that, system SW such as device drivers had to be adapted to support multiple FPGA devices attached to an ARM-based CPU. Moreover, we developed a communication mechanism that allows multi-FPGA execution via message passing in a similar fashion as the Message Passing Interface (MPI) called OmpSs MPI for FPGAs (OMPIF) [7]. This not only enables direct communication between FPGA devices in the same node, which is critical in a multi-FPGA application but enables direct communications between FPGAs in different nodes. We developed a simple API to implement communications inside the accelerators. Basic API calls are shown in listing 1.

```
void OMPiF_Send(const void* buf, int count,
               OMPiF_Datatype datatype, int dest, int tag,
               OMPiF_Comm comm);
void OMPiF_Recv(void* buf, int count,
               OMPiF_Datatype datatype, int dest,
               int tag, OMPiF_Comm comm);
```

Listing 1: OMPiF API calls.

Moreover, the APEIRON framework [8] is used as a backend to implement low-level data transmission and reception for the IDV-E platform. Raw 100G Ethernet is also supported as a communication backend.

D. Hardware IPs for low-latency Inter-FPGA communication

The INFN Communication IP is the main component of the APEIRON framework, allowing direct communication between tasks deployed on the same FPGA (intra-node communication) and on different FPGA (inter-node communication),

without involving CPU and system bus resources. It is based on the HPC direct network designs previously developed by INFN APE Lab, *i.e.* APENet [9] and ExaNet [10][11].

As shown in fig. 2, the Communication IP can be split into a *Network IP* and a *Routing IP* block. The *Network IP* defines the data encoding scheme for the messages over the cables; it implements inter-node ports to transfer data between neighbor FPGAs using AMD Aurora 64B/66B cores and 10G/25G Ethernet supporting UDP/IP transport layer offloading. Meanwhile, *Routing IP* manages data transfers between Communication IP ports, applying the dimension-order routing policy for inter-node communications and solving contentions between packets reaching the same port.

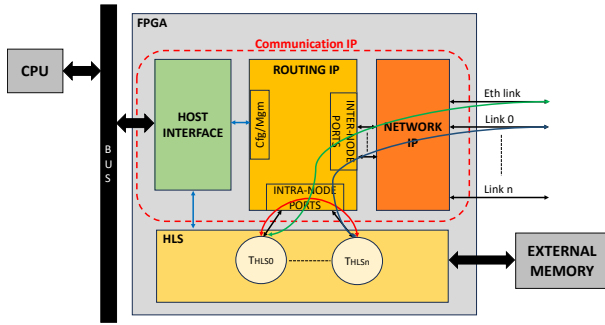


Fig. 2: Block diagram of INFN communication IP showing intra-node (red) and inter-node (green blue) communications.

In the testbench, the Communication IP featured with 4 intra-node ports and 2 inter-node ports, is implemented as an RTL-IP kernel connected to the global system/board clock of 150 MHz and to 4 dispatcher/aggregator couples.

In latency tests, a *send_receive* HLS kernel in the *initiator FPGA* reads a payload (of max 4096 Bytes) from the memory (either BRAM or DDR), sends it to a destination (*pipe kernel*) and waits for an acknowledge packet from the pipe kernel. To minimize the host call overhead, one million *send_receive* operations are launched. The time elapsed from the start of the first packet sent to the completion of the last packet received is measured on the host. Based on the destination, we performed three latency tests: destination task deployed on the same FPGA and intra-node port of sender (local-loop), different intra-node port (local trip), and a different FPGA (roundtrip).

In fig. 3a it's possible to notice the effects of the DDR memory access latency and synchronization overheads between the CPU and FPGA. Using the BRAM, the end-to-end latency remains below 1 μ s for packet payload sizes up to 512B.

Bandwidth test is carried out by transferring multiple data packets with fixed payload size from a *sender* HLS kernel and receiving a single *ACK* packet to confirm the reception. According to source and destination, we performed *Loopback* test (on the same FPGA) and *Oneway* test (different FPGAs). As shown in fig. 3b), bandwidth does not saturate at the maximum message size (4KB). Moreover, DDR latency dominates the transfer cost as bandwidth is the same regardless if source and destination being or not in different FPGAs.

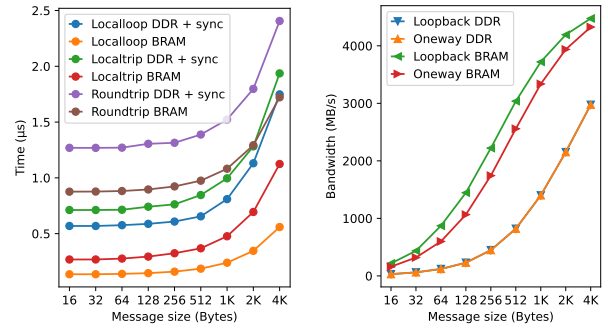


Fig. 3: Measured latency (a) and bandwidth (b) using 256-bit internal datapath width for communication IPs.

E. The APEIRON multi-FPGA stream-based framework

The APEIRON framework has been developed to offer HW and SW support for running real-time dataflow applications on a multi-FPGA system. It implements the following features:

1) *Automatic project linking and bitstream generation*: Starting from a user-defined YAML configuration file which describes the attributes of each HLS kernel (number of input and output channels, its dedicated IntraNode port), APEIRON framework can create bitstream which implements a design including the Communication IP and the HLS kernels, kernels, along with their arguments, must expose a generic AXI4-Stream interface for each communication channel as shown in listing 2.

```
void example_apeiron_task(
    [optional kernel-specific list of parameters],
    message_stream_t message_data_in[N_IN_CHANNELS],
    message_stream_t message_data_out[N_OUT_CHANNELS]);
```

Listing 2: HLS kernels prototype.

2) *HLS Inter-FPGA communication library*: The communication between kernels is expressed via HAPECOM: a lightweight C++ API, based on non-blocking *send()* and *receive()* operations. This simple API allows the HLS developer to perform communications between kernels, deployed on the same or different FPGAs, without knowing the details of the underlying packet communication protocol.

```
size_t send(msg, size, dest_node, task_id, ch_id);
size_t receive(ch_id);
```

Listing 3: HAPECOM API pseudocode.

A sample of the API calls is shown in listing 3 where:

- *dest_node* is the n-Dim coordinate of the destination node (FPGA) in an n-Dim torus network;
- *task_id* is the local-to-node receiving task (kernel) identifier (0-3);
- *ch_id* is the local-to-task receiving FIFO (channel) identifier (0-127).

The Communication Library uses AXI4-Stream Side Channels to encode all the information needed to build the packet header. Then, an *Aggregator* IP receives outgoing packets from the task and builds the packet header and a *Dispatcher* IP receives incoming packets from the Routing IP and forwards them to the right kernel input channel.

3) *Runtime host sw stack and monitoring tools*: The host SW stack provides runtime support for the multi-FPGA execution model. It is based on AMD *xocl* and *XCLMGMT* drivers used in combination with Xilinx Runtime library (XRT), an open-source SW stack that facilitates the management and usage of FPGA/ACAP devices.

The SW stack is composed of different modules. The *Apeironlib* module provides user access to low-level XRT functionality. It also manages access from different processes to the same FPGA board. *Apeirond* is a network-exposed server used to manage multiple (local or remote) access requests from user apps to an FPGA. It has a persistent handler over the FPGA board which is an instance of the *Apeironlib* module. It operates on the client/server principle: it listens for user application requests over the network.

The *Apeiron*s component is responsible for the connection handling and the request parsing, exposing *Apeironlib* commands over the network. The communication protocol uses JSON messages over TCP/IP sockets. The *Monitoring tools* are used to monitor the status of the nodes in the network from a single node running the *Supervisor* module. They provide a graphical view of the status of all nodes in the cluster. The *Supervisor* module operates remotely on the target nodes by flashing bitstreams, running kernels and writing and reading registers and execution logs. Information such as kernel state (running, stopped, idle, etc.), power consumption, and thermal information is gathered for each board in each node.

F. Precision Tuning

Precision tuning is an approximate computing technique that trades off the accuracy of the result for improvements in performance and latency [12]. Several tools and techniques have been studied in the literature, employing either static analysis or code profiling to understand the trade-offs, and then, compiler transformations to effectively alter the precision of the computation by replacing instructions and data types. It is worth noting that switching between different precision levels, particularly when they are achieved through different number systems representations for real numbers (e.g., fixed point and floating point), has a non-negligible cost that must be accounted for in the trade-off [12]. In TEXTAROSSA, precision tuning has been explored through the TAFFO [13], [14] tool, a set of plugins for the LLVM compiler framework [15].

In particular, TEXTAROSSA proposes three extensions to TAFFO: (i) support for GPGPU architectures; (ii) support for High-Level Synthesis tools; (iii) support for the Posit number system. The latter extension broadens the support of TAFFO for different number systems. Posit number system support was added on top of fixed and floating point. This can be useful since Posit has variable precision, that

allows more accurate representations at a lower bit size [16]. The first two extensions, on the other hand, help target the IDVs of the TEXTAROSSA project. In the GPGPU case, the key extension is the support for heterogeneous computing platforms, enabling the separate but coherent compilation of host- and device-side code. It can be proven experimentally that supporting precision tuning without keeping aligned the data types across the heterogeneous platform, results in a massive overhead due to undue type conversions, as the two architectures very likely have different optimal types when considered in isolation [17]. Finally, the main difficulty of integrating high-level synthesis tools is related to the lack of support from VitisHLS, which composes the backbone of the TEXTAROSSA reconfigurable computing toolchain, of a recent version of the LLVM compiler framework. Due to the rapid evolution of the compiler infrastructure, to which VitisHLS does not keep aligned (it is based on version 8 of LLVM, whereas the latest LLVM major release is version 18) there was a need to backport TAFFO to the earlier major version of LLVM, compatible with VitisHLS.

IV. EVALUATION AND ACHIEVEMENTS

A. Multi-node FPGA support

We used the n-body simulation application to evaluate the *OmpSs@FPGA* multi-node task-based programming model, task scheduling IP, and the APEIRON communications infrastructure. We run this benchmark using the IDV-E platform as well as the Meep machine, to further evaluate the scalability of the proposed solution. The Meep system [18] is composed of 12 nodes with 8 AMD Alveo U55c FPGAs in each node. FPGAs are AMD Alveo U55c, which feature the same FPGA chip as the IDV-E but with a different memory topology. Fig. 4 shows a summary of the performance obtained for the n-body application in different scenarios.

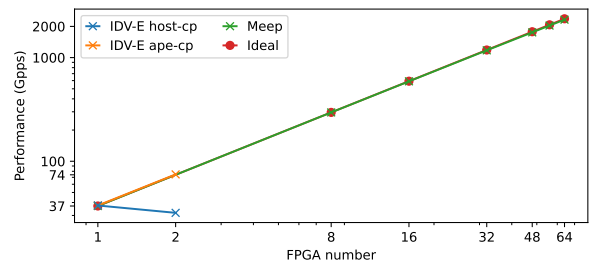


Fig. 4: N-body performance vs. number of FPGAs.

The inter-accelerator interconnect allows accelerators to exchange data without host intervention. This allows the application to scale beyond a single FPGA device as shown for the case of *IDV-E ape-cp* line in fig. 4, otherwise, data copies between the host and accelerators quickly become a bottleneck causing performance degradation as shown in the *IDV-E host-cp* line. Moreover, by using a HW module to schedule tasks, we can manage a large number of accelerators as shown in the *Meep* line of fig. 4, which is very close to the

ideal performance even for 64 FPGAs. All in all, we can reach 74.7 Gpps (billions of particle pairs processed per second) in the IDV-E system and 2322.98 Gpps in the larger meep system. Regarding energy efficiency, we reach 0.39 Gpps/W in the 2-phase cooled IDV-E, 0.35 Gpps/W in the air-cooled IDV-E, and 0.36 Gpps/W in the air-cooled Meep. This highlights that cooling technology has a non-negligible effect on power efficiency.

B. The RAIDER Multi-FPGA Inference Application

RAIDER is a high-throughput online streaming processing application implemented on FPGA. Its task is to perform Particle IDentification (PID) on the stream of events generated by the RICH (Ring Imaging CHerenkov) detector in the CERN NA62 experiment, using Convolutional Neural Networks (CNN). This CNN model has been tested and trained offline using Tensorflow/Keras and then deployed on FPGA with the HLS4ML [19] tool. The CNN input data is a 16x16 image for each RICH physics event and produces, as output prediction, an estimate for the number of charged particles in a single event by doing a classification between 4 output classes: 0, 1, 2, or 3+ rings. Since this implementation has to cope with the 10 MHz event rate of the NA62 L0 trigger, sustaining an adequate processing throughput is the main challenge for such a system. However, the initiation interval of the CNN obtained from HLS4ML increases with the size of the image, reducing the processing timing performance. Thus, to improve the throughput, the RAIDER application has been designed using the APEIRON framework with multiple processing kernels placed in different nodes, capable of receiving events coming from the network via the HAPECOM communication APIs. In this setup, the interconnected boards are used as nodes of a RAIDER deployment via the APEIRON framework with distinct roles:

- *Preprocessing node*: data is loaded from Host memory and sent through the network via an HLS kernel (krnl_sender). Data is then processed by 3 Imagifier HLS kernels which convert the PMT hitlist information into a 256-bit word (16x16 B&W image) that is sent to the Computing node through the internode ports of the INFN Communication IP. As a second task, this node is in charge of receiving the output of the CNN computation and storing it on Host memory via an HLS kernel (krnl_receiver).
- *Computing node*: images coming from external nodes are taken as input and dispatched to various CNN HLS kernels (each of them connected to a different INFN Communication IP intranode port) to compute the predictions. Results are then sent back to the preprocessing node.

Fig. 5 shows how these components are laid out across multiple nodes and their interconnection paths between them.

RAIDER clustering quality measured as sensitivity (% data points correctly classified) and precision (% of correct data points in a class) are reported in Table I. They were measured by taking 2.7M events, extracted from the NA62 database,

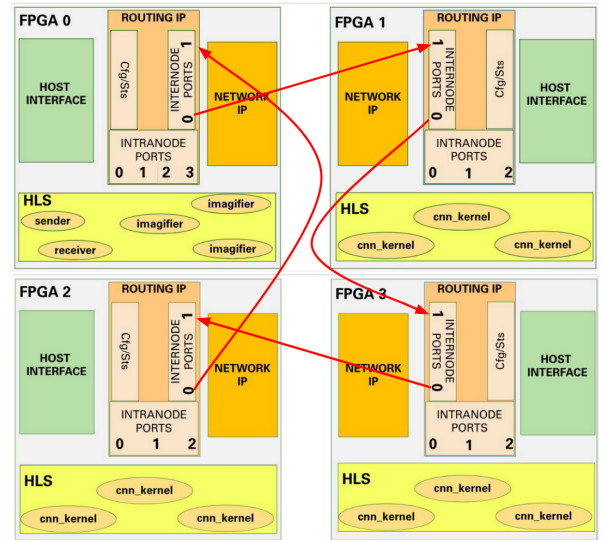


Fig. 5: Test setup of 4 AMD Alveo U280 boards installed on IDV-E nodes.

Class	Sensitivity	Precision
0	92%	83%
1	79%	88%
2	75%	70%
3+	76%	80%

TABLE I: Efficiency and purity values obtained from FPGA implemented CNN model trained on NA62 physical events number of rings classification.

as neural network input. To test the peak throughput and energy efficiency values of the application, we decided to work with a subset of events (sent through the network for 2.7M times) loaded from the BRAM instead of working with the whole NA62 dataset loaded on the DDR FPGA memory. This is due to a limit in the *sender* HLS kernel, which needs ~ 160 clock cycles to load events from DDR memory, limiting maximum throughput to 1.278 MHz. We run this application on two different setups: four interconnected AMD Alveo U280 installed on the IDV-E node in a ring topology, 2 FPGAs for each of the two IDV-E nodes, and four interconnected AMD Alveo U200 installed in the INFN Roma APE Lab in a ring topology, one FPGA for each of the four single Intel Xeon Silver 4410T CPU nodes. All tests performed in both testbeds have been done using a 200 MHz global clock in the HW setups and by scaling the number of CNN HLS processing kernels starting from tests on 2 nodes (one preprocessing and one computing) up to 4 nodes (adding 2 more computing nodes).

To evaluate RAIDER processing throughput, we tested the system scaling from 2 FPGAs (one preprocessing and one computing) up to 4 (adding 2 more computing FPGAs). However, since the resources available on the Alveo U280s are larger than the Alveo U200 FPGAs of the previous testbed, we implemented 3 CNNs HLS kernels on each computing FPGA. Both throughput and number of CNN kernels are reported in

table II. To compare and visualize the trends of the throughput

APE (U200)		IDV-E (U280)	
#CNN	Throughput (Mevents/s)	#CNN	Throughput (Mevents/s)
2	1.163	3	1.813
4	2.325	6	3.409
6	2.692	9	4.874

TABLE II: Processing throughput with an increasing number of CNN HLS kernels on Xilinx Ulveo 200 APE Lab testbed and IDV-E Alveo U280

values reported in the Table II under *APE* and *IDV-E* for the APEIRON setup and the IDV-E setups respectively, we choose to picture them in fig. 6. In detail, from the linear regression curve obtained from the different sets of results, we can notice that the RAIDER application tends to scale better on the U280-based IDV-E testbed, since in these computing nodes more CNN HLS kernels can be implemented, more precisely, 3 CNN kernels can be placed in each FPGA instead of the 2 that fit in the U200 device increasing the global computing performance of the HW.

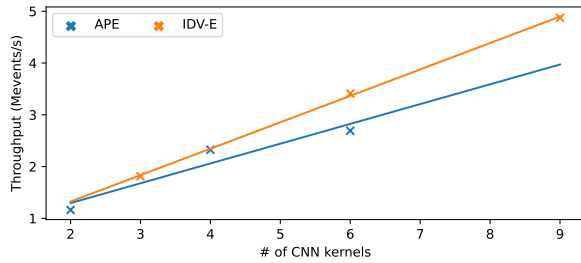


Fig. 6: RAIDER Throughput scaling trends for the IDV-E Alveo U280 setup (blue) and Xilinx U200 (orange).

C. Simulating quantum systems on IDV-A

Quantum TEA is among the applications studied within Testarossa. Simulations of quantum systems face an exponential scaling of resources when adding more particles; tensor network methods avoid this by compressing quantum correlations in a tunable parameter χ . The Quantum TEA library implemented flexible precisions and GPU support during the TEXTAROSSA project. As a benchmark problem on IDV-A, we cool a one-dimensional quantum Ising model all the way down to the ground state via a variational tree tensor network algorithm. We choose 64 qubits, $\chi = 64$, and four sweeps of patterns like SSSD (Z=double complex, D=double, S=single). Errors are available via an analytic solution. The "mixed precision" approach increases the precision during the sweeps with significant speedup at equal error: 272s \rightarrow 117s \rightarrow 39s for ZZZZ (only option at project start), DDDD, SSSD sweep patterns, respectively (qtealeaves-numpy). At this moderate bond dimension, one can already see a benefit of GPUs for complex arithmetic with 36s \rightarrow 13s for switching from the fastest CPU code to the best GPU while decreasing the error

by two orders of magnitude (qgreentea-fortran \rightarrow qtealeaves-torch-gpu). These improvements have an actual impact on how this type of simulation is approached in the future.

D. UrbanAir HPC application

UrbanAir is a multiscale HPC application to predict air quality in urban environments. Detailed prediction at city or street level requires running over a domain with 10m horizontal resolution, starting from a 50M gridpoints domain. This is a challenge to solve such problems at this fine-grain scale on traditional HPC systems in a reasonable amount of time. Another aspect being considered is to maximize energy efficiency while shortening time-to-solution. Another limitation comes from the memory available for GPUs, preventing to solve problems on very large domains on a single GPU. Therefore, implementation on multi-GPU is required which exploits IDV-A most efficiently. In TEXTAROSSA, focus is given to the GCRK routine of UrbanAir, which is a preconditioner with an iterative solver. The toolchains used by UrbanAir include the C++ compiler, the CUDA toolkit, OpenMP for shared-memory parallelization (single node), and MPI for data exchange between GPUs, all available at the IDV-A platform. To monitor the energy consumption of kernels running on GPUS, GPowerU project tool was used, also available on the IDV-A TEXTAROSSA platform. Fig. 7 presents energy usage when all GPUs available at the node are taken into consideration, i.e. for a single GPU run, energy usage of all 4 GPUs available is summed. *Dibona* is the former IDV-A system, while on final IDV-A, three different configurations are compared: IDV-A with traditional cooling system (*wo-2phase*), IDV-A with two-phase cooling system installed (*w-2phase*), IDV-A with two-phase cooling system on which optimized version of UrbanAir-gcrk was run (*w-2phase-opt*). The most energy-efficient execution is when all available GPUs are used for computations. The installation of 2-phase cooling systems has a small impact on application performance (see fig. 8) and energy efficiency. Both are improved with the introduced optimizations.

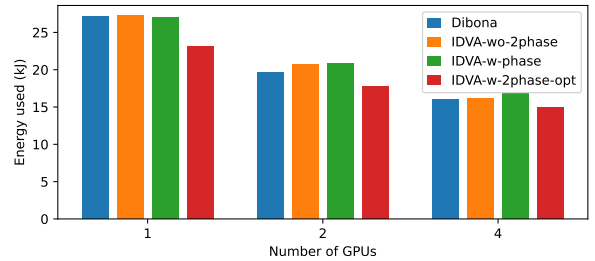


Fig. 7: UrbanAir-gcrk energy consumption for 59M grid points, unused GPUs accounted.

E. Math Library

Some modules of a mathematical library for GPU-accelerated heterogeneous architectures, such as IDV-A, have been developed and tested during the project. They include

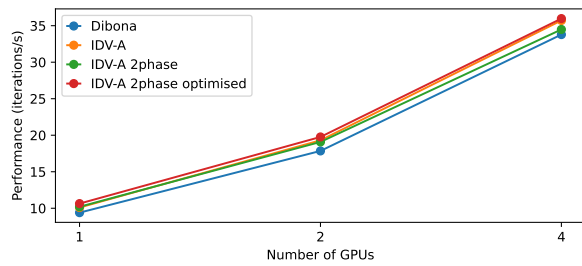


Fig. 8: UrbanAir-grk iterations/s for 59M grid points.

computational kernels required in sparse matrix computations and iterative linear solvers, which are widely used in HPC and HPDA/AI domains, and, as well known, are memory/communication bound modules. Main efforts were put on GPU-kernel efficiency and scalability in multiple GPUs. Multiple GPUs are often needed because dimensions exceed the memory resources of a single GPU. Therefore, on IDV-A, the library stressed the GPU operation capabilities and memory/communication channel bandwidth at the node level. The library development toolchain includes C compilers, the Cuda Toolkit, and the MPI library available as the basic environment of the TEXTAROSSA platform. These tools allow reusing very efficient GPU kernels for single Nvidia GPUs previously developed and focus on algorithms that enable scalability on a large number of GPUs. Moreover, extensive use of the GPowerU project tool, developed by INFN, has been made for monitoring GPU kernel energy consumption. Details on the algorithms and parallel design patterns implemented for all kernels of the library are widely discussed in [20].

For the main solver, *BCMGX*, we can reach a performance 2.17×10^6 DoFs/s (degrees-of-freedom per second) using 4 GPUs for a problem size of 244M DoFs in IDV-A. Two-phase cooling reduces energy to solution by reaching a maximum efficiency of 19.5×10^4 DoFs/J.

V. CONCLUSIONS

The project serves as a prime example of successful cooperation among the various partners, with many of its outcomes positioned for further use in other research avenues and industrial applications. Notably, a commercial computing system utilizing evaporative cooling and featuring blades manufactured by E4, with a design akin to IDV-A, has been successfully installed as of March 2024 within the computing center hosted by the University of Turin. The impact of such cooling techniques has proven to be relevant to achieve high energy efficiency. Moreover, applications, tools, and HW accelerators have been successfully adapted to support heterogeneous multi-node systems, providing scalability across clusters with a large number of nodes.

ACKNOWLEDGEMENTS

This work is supported by the (EuroHPC) TEXTAROSSA project G.A. n.956831, by the Spanish Government (Grants PCI2021-121964 - TEXTAROSSA, PID2019-107255GB-C21

MCIN/AEI/10.13039/501100011033, and CEX2021-001148-S), by Generalitat de Catalunya (2021 SGR 01007), by the Italian National Resilience and Recovery Plan (PNRR) via the National Center for HPC, Big Data and Quantum Computing, by the Italian Departments of Excellence grant 2023-2027 Quantum Frontiers, by the Italian Ministry of Economic Development (MISE) project n. 2774 "TEXTAROSSA", the European Union via projects EuRyQa and QuantEra's T-NISQ, and the German Federal Ministry of Education and Research (BMBF) via QRydDemo.

REFERENCES

- [1] G. Massari *et al.*, "Reliability-oriented resource management for high-performance computing," *Sustainable Computing: Informatics and Systems*, vol. 39, p. 100873, 2023.
- [2] W. Fornaciari *et al.*, "The textarossa approach to thermal control of future hpc systems," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Cham: Springer, 2022, pp. 420–433.
- [3] F. Terraneo *et al.*, "3D-ICE 3.0: efficient nonlinear MPSoC thermal simulation with pluggable heat sink models," *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.
- [4] A. Leva *et al.*, "Event-based power/performance-aware thermal management for high-density microprocessors," *IEEE Trans on Control Systems Technology*, vol. 26, no. 2, pp. 535–550, 2018.
- [5] L. Morais *et al.*, "Enabling hw-based task scheduling in large multicore architectures," *IEEE Transactions on Computers*, vol. 73, no. 1, pp. 138–151, 2024.
- [6] A. Filguera *et al.*, "Fpga framework improvements for hpc applications," in *2023 International Conference on Field Programmable Technology (ICFPT)*, 2023, pp. 286–287.
- [7] J. M. de Haro *et al.*, "Ompss@cloudfpga: An fpga task-based programming model with message passing," in *IPDPS'22*, 2022, pp. 828–838.
- [8] Ammendola, Roberto *et al.*, "Apeiron: A framework for high level programming of dataflow applications on multi-fpga systems," *EPJ Web of Conf.*, vol. 295, p. 11002, 2024.
- [9] R. Ammendola *et al.*, "Apenet+ 34 gbps data transmission system and custom transmission logic," *Journal of Instrumentation*, vol. 8, no. 12, p. C12022, dec 2013.
- [10] —, "Large scale low power computing system: Status of network design in exanest and euroexa projects," *Advances in Parallel Computing*, vol. 32, pp. 750–759, 2018.
- [11] M. Katevenis *et al.*, "Next generation of exascale-class systems: Exanest project and the status of its interconnect and storage development," *Microprocessors and Microsystems*, vol. 61, pp. 58 – 71, 2018.
- [12] S. Cherubin *et al.*, "Tools for reduced precision computation: a survey," *ACM Computing Surveys*, vol. 53, no. 2, Apr 2020.
- [13] —, "Dynamic precision autotuning with taffo," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 2, May 2020.
- [14] D. Cattaneo *et al.*, "Architecture-aware precision tuning with multiple number representation systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 673–678.
- [15] C. Lattner *et al.*, "LLVM: A compilation framework for lifelong program analysis & transformation," in *CGO'04*, ser. CGO '04. IEEE Computer Society, 2004, p. 75.
- [16] J. L. Gustafson *et al.*, "Beating floating point at its own game: Posit arithmetic," *Supercomputing Frontiers and Innovations*, vol. 4, no. 2, pp. 71–86, 2017.
- [17] D. Cattaneo *et al.*, "Mixed precision in heterogeneous parallel computing platforms via delayed code analysis," in *International Conference on Embedded Computer Systems*. Springer, 2023, pp. 469–477.
- [18] E. Perdomo *et al.*, "Makinote: An fpga-based hw/sw platform for pre-silicon emulation of risc-v designs," in *Proceedings of the 16th Workshop on Rapid Simulation and Performance Evaluation for Design*, ser. RAPIDO '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 29–34.
- [19] J. Duarte *et al.*, "Fast inference of deep neural networks in FPGAs for particle physics," *JINST*, vol. 13, no. 07, p. P07027, 2018.
- [20] M. Bernaschi *et al.*, "A multi-GPU aggregation-based AMG preconditioner for iterative linear solvers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 8, pp. 2365–2376, 2023.