Microprocessors and Microsystems 122 (2026) 105254

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

The TEXTAROSSA project: Cool all the Way Down to the Hardware☆

Antonio Filgueras [a,b,*], Giovanni Agosta [c,l], Marco Aldinucci [m,p], Carlos Álvarez [a,b], Pasqua D'Ambra [m], Massimo Bernaschi [m], Andrea Biagioni [d], Daniele Cattaneo [c], Alessandro Celestini [m], Massimo Celino [f], Carlotta Chiarini [d,n], Francesca Lo Cicero [d], Paolo Cretaro [d], William Fornaciari [c,l], Ottorino Frezza [d], Andrea Galimberti [c], Francesco Giacomini [o], Juan Miguel de Haro Ruiz [a,b], Francesco Iannone [f], Daniel Jaschke [j,i,h], Daniel Jiménez-González [a,b], Michał Kulczewski [e], Alberto Leva [c], Alessandro Lonardo [d], Michele Martinelli [d], Xavier Martorell [a,b], Simone Montangero [h,k,i,j], Lucas Morais [a,b], Ariel Oleksiak [e], Paolo Palazzari [f], Luca Pontisso [d], Federico Reghenzani [c,l], Cristian Rossi [d,n], Sergio Saponara [g,l], Carlo Saverio Lodi [c], Francesco Simula [d], Federico Terraneo [c,l], Piero Vicini [d], Miquel Vidal [b], Davide Zoni [c], Giuseppe Zummo [f]

[a] Universitat Politècnica de Catalunya, Spain
[b] Barcelona Supercomputing Center, Spain
[c] DEIB – Politecnico di Milano, Italy
[d] INFN, Sezione di Roma, Italy
[e] Poznańskie Centrum Superkomputerowo-Sieciowe, Poland
[f] ENEA, C.R. Casaccia, Via Anguillarese 301, Rome, Italy
[g] University of Pisa, Italy
[h] Dipartimento di Fisica e Astronomia "G. Galilei", Università degli Studi di Padova, Italy
[i] INFN, Sezione di Padova, Italy
[j] Institute for Complex Quantum Systems, Ulm University, Germany
[k] Padua Quantum Technologies Research Center, Italy
[l] Centro Interuniversitario Nazionale per l'Informatica (CINI), Italy
[m] National Research Council (CNR), Italy
[n] Università "Sapienza" di Roma, Italy
[o] INFN, CNAF, Italy
[p] University of Torino, Italy

ARTICLE INFO

Keywords:
European project
Heterogeneous architectures
High-performance computing
Energy Efficiency
Cooling techniques

ABSTRACT

The TEXTAROSSA project aims to bridge the technology gaps that exascale computing systems are currently facing and will be key in the near future to overcome performance and energy efficiency challenges. This project provides solutions for improved energy efficiency by using state-of-the-art cooling and thermal control, seamless integration of heterogeneous accelerators in HPC multi-node platforms, and new arithmetic methods tailored to heterogeneous hardware platforms. 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.

☆ This article is part of a Special issue entitled: 'EPESD 2025' published in Microprocessors and Microsystems.

* Corresponding author at: Universitat Politècnica de Catalunya, Spain.
E-mail addresses: antonio.filgueras@bsc.es (A. Filgueras), giovanni.agosta@polimi.it (G. Agosta), aldinuc@di.unito.it (M. Aldinucci), carlos.alvarez@upc.edu (C. Álvarez), pasqua.dambra@cnr.it (P. D'Ambra), massimo.bernaschi@cnr.it (M. Bernaschi), andrea.biagioni@roma1.infn.it (A. Biagioni), daniele.cattaneo@polimi.it (D. Cattaneo), alessandro.celestini@cnr.it (A. Celestini), massimo.celino@enea.it (M. Celino), carlotta.chiarini@roma1.infn.it (C. Chiarini), francesca.locicero@roma1.infn.it (F.L. Cicero), paolo.cretaro@roma1.infn.it (P. Cretaro), william.fornaciari@polimi.it (W. Fornaciari), ottorino.frezza@roma1.infn.it (O. Frezza), andrea.galimberti@polimi.it (A. Galimberti), francesco.giacomini@cnaf.infn.it (F. Giacomini), juan.deharoruiz@bsc.es (J.M.d.H. Ruiz), francesco.iannone@enea.it (F. Iannone), daniel.jaschke@gmail.com (D. Jaschke), daniel.jimenez-gonzalez@upc.edu (D. Jiménez-González), michal.kulczewski@man.poznan.pl (M. Kulczewski), alberto.leva@polimi.it (A. Leva), alessandro.lonardo@roma1.infn.it (A. Lonardo), michele.martinelli@roma1.infn.it (M. Martinelli), xavier.martorell@upc.edu (X. Martorell), simone.montangero@unipd.it (S. Montangero), morais.lucas.h@gmail.com (L. Morais), ariel@man.poznan.pl (A. Oleksiak), paolo.palazzari@enea.it (P. Palazzari), luca.pontisso@roma1.infn.it (L. Pontisso), federico.reghenzani@polimi.it (F. Reghenzani), cristian.rossi@roma1.infn.it (C. Rossi), sergio.saponara@unipi.it (S. Saponara), carlo.saverio.lodi@polimi.it (C.S. Lodi), francesco.simula@roma1.infn.it (F. Simula), federico.terraneo@polimi.it (F. Terraneo), piero.vicini@roma1.infn.it (P. Vicini), miquel.vidal@bsc.es (M. Vidal), davide.zoni@polimi.it (D. Zoni), g.zummo@in-quattro.com (G. Zummo).

https://doi.org/10.1016/j.micpro.2026.105254
Received 14 March 2025; Received in revised form 26 September 2025; Accepted 12 February 2026
Available online 28 February 2026

1. 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 increasingly leaning 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] CINI, an Italian consortium grouping together three leading universities, Politecnico di Milano, Università degli Studi di Torino, and Università di Pisa, Fraunhofer (Germany), INRIA (France), ATOS (France), E4 Computer Engineering (Italy), BSC (Spain), PCSS (Poland), INFN (Italy), CNR (Italy), In Quattro (Italy), Université de Bordeaux (France), CINECA (Italy), and Universitat Politècnica de Catalunya (Spain). The three Italian universities are part of the lab of CINI,[3] created in 2021, that is grouping the main academic and research entities working in the field of high-performance and Exascale computing in Italy. CINI is also providing the technical leadership of the project. This project builds on the results of previous projects: RECIPE [1], AXIOM [2], LEGaTO [3], EuroEXA [4], EPEEC [5], INTERTWinE [6], EXA2PRO [7], ExaNoDe [8], ExaNeSt [9], MANGO [10], ANTAREX [11], ASPIDE [12], VECMA [13], and COMPAT [14]. These projects serve as a foundation on which tools and techniques are built upon, as envisioned on the project inception [15].

More information on the activities carried out during the execution of TEXTAROSSA can be found in the project website.[4] The paper is organized as follows. Section 2 describes the HW platforms designed during the project. Section 3 describe project contributions to the HW/SW stack. Section 4 presents evaluation of the contributions across different use cases. Finally, Section 5 draws some conclusions.

2. Hardware platforms

During the project, we developed two 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 2.1 and 2.2, respectively. Both these platforms utilize the project-designed two-phase cooling technology described in Section 3.1. Although both platforms use off-the-shelf components and adhere to standards such as OpenSequana, which are widely adopted in the HPC domain, the cooling system installations have not yet been engineered to minimize vertical space usage, as they are still at the prototype stage. In order to evaluate some of the proposed solutions in larger-scale scenarios, we used production-grade HPC systems. Alternative platforms similar to IDV-A and IDV-E were also used at early development stages.

For the GPU, Dibona and Altair were used. Altair is an HPC machine operated by Poznańskie Centrum Superkomputerowo-Sieciowe (PCSS). The GPU part consists of 9 nodes, each equipped with Intel Xeon Gold 6242 2.8 GHz (2 × 16 CPU cores), 8 NVIDIA V100 SXM2 GPUs, 384 GB RAM, and Infiniband EDR/NDR. Dibona is the former IDV-A, a BullSequana XH2000 platform providing one CRRM blade with no connection to high-speed interconnect, and the only access is a 1 Gb/s link to the CPU host. The CRRM blade is composed of dual AMD EPYC 7402P with 24 cores and 48 threads per CPU, and a total of 512 GB of main memory. This node contains 4 NVIDIA A100 GPU XSM-40 GB GPU. They are attached via the NVIDIA SXM4 socket. This provides PCIe Gen 4 x16 connectivity to the host system as well as 4 NVLink lanes to each of the other GPUs.

For FPGA, we tested the scalability of proposed solutions using the Meep platform [16], which contains 96 AMD Alveo U55c FPGAs across 12 dual Intel Xeon Gold 6330 processors with 256 GB of main memory. FPGAs are attached to the host system via PCIe Gen 3 x16 and are connected using a 100G Ethernet network, which is independent of the main host-attached 100G Ethernet network. Also, the APE platform was used in the early development stage. It is composed of four AMD Alveo U200 FPGAs installed in four Intel Xeon Silver 4410 T based nodes and connected using PCIe Gen 3 X16. Energy efficiency comparisons were made with Dibona for the IDV-A platform. Marenostrum 4 [17], which contains dual Intel Xeon Platinum 8160 nodes with 96 GB of main memory and connected using OmniPath, as well as Meep, are used to provide power efficiency comparisons for the FPGA-based IDV-E.

2.1. IDV-A

The IDV-A prototype is based on the 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.

2.2. 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 them. 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.

3. Project contributions

3.1. 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 [18,19]. 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 boiling fluid 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.

[1] https://eurohpc-ju.europa.eu/.

[2] https://textarossa.eu/consortium/ (last accessed March 2025).

[3] https://www.consorzio-cini.it/index.php/it/laboratori-nazionali/hpc-key-technologies-and-tools (last accessed March 2025).

[4] https://textarossa.eu (last accessed March 2025).

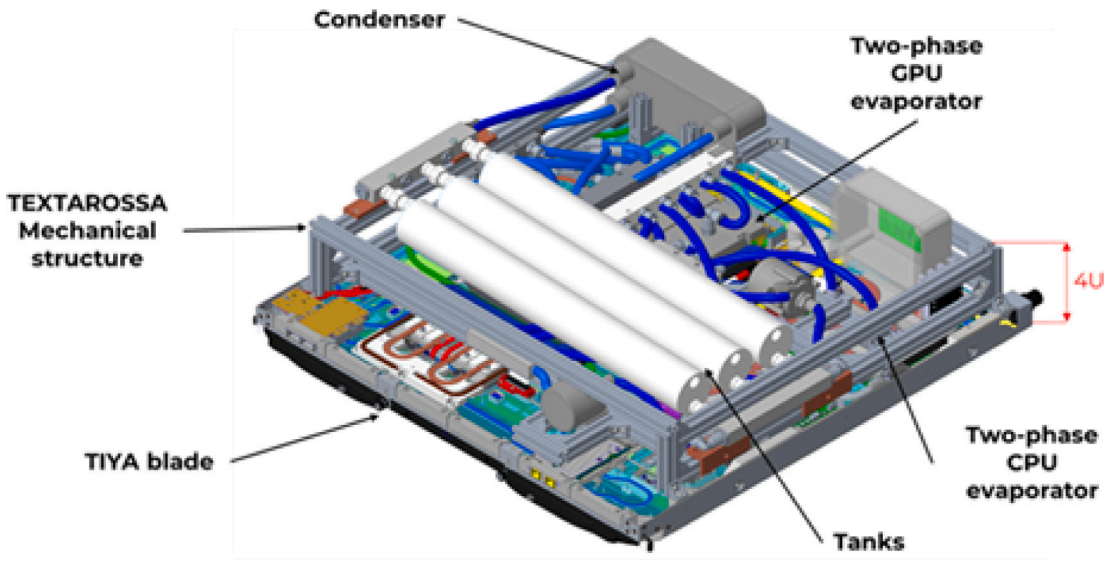


Fig. 1. Evaporative cooling installed on IDV-A.

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 [20]. 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 2, showing the relevant parts of the two-phase cooling solution.

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 cosimulation with the 3D-ICE chip thermal simulator [21]. The goal of a thermal model is to accurately reproduce 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. A 10 240 × 10 240 × 625 µm silicon chip is simulated in 3D-ICE with a spatial discretization set to a resolution of 32 × 32 µm in the x and y coordinates, while vertically, a 15 µm layer is used to simulate the active silicon where power is dissipated, followed by a 610 µm layer representing the silicon bulk. 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 × 320 × 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 the creation of fine-grain thermal maps of the surface of the chip, enabling fine-grain control over the parameters. Maintaining the status of phase change of the coolant is not a trivial task: optimal performance occurs when both the flow rate and the dissipated power are either high or low simultaneously. Fig. 2 shows, in a simplified manner, that there exists only

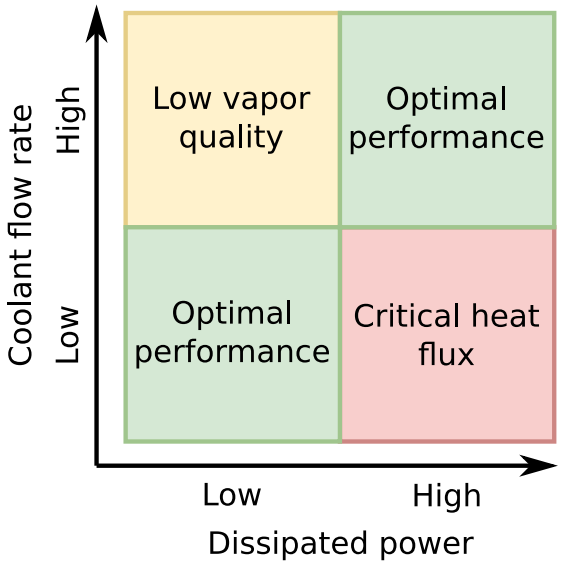


Fig. 2. Operating modes of the cooling system.

a limited amount of the operating region with optimal performance of the 2-phase cooling.

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 CPU architectures, 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. The evaporative coolant system has completely different dynamics compared to

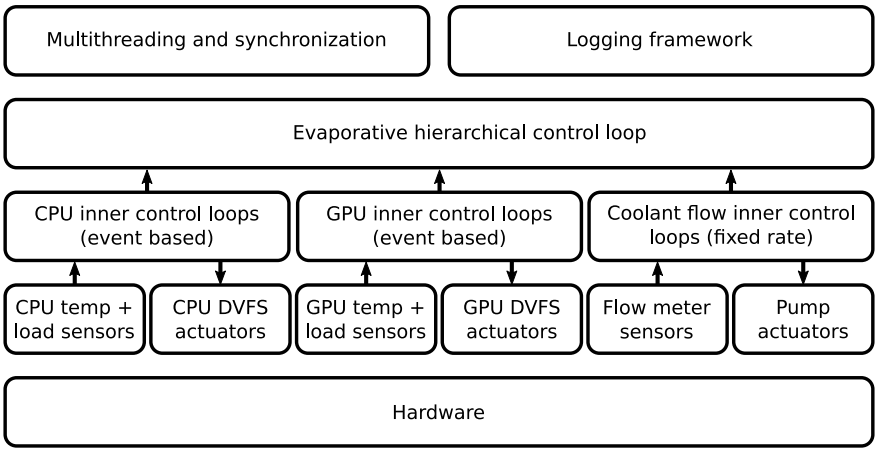


Fig. 3. Software architecture of the thermal controller.

the CPU. Due to the involved thermal inertia, the evaporative cooling system is considerably slower compared to the on-chip phenomena. Moreover, the pump actuators exhibit a response speed comparable with fans, and therefore considerably slower than DVFS. The thermal controller needs to be designed taking these physical limitations into account. 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 hierarchically splits the control problem into one fast inner control loop per controlled device (CPU core/GPU), plus one evaporative controller per cooling circuit, 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 [22], and made available to the Textarossa project as an IP [22]. As the thermal control policy is implemented in software and runs on the same computing devices it is controlling, it competes for resources with the running applications. It is thus important to reduce the overhead of the control policy as much as possible. The proposed control strategy utilizes event-based control theory to reduce the overhead caused by the thermal controller, by running the control algorithm only when needed, instead of doing so periodically as would happen with classical control theory [22]. The overall architecture of the C++ based implementation of the thermal controller is depicted in Fig. 3.

The purpose of the hierarchical controller is to dynamically set the coolant flow rate based on the temperature and thus workload conditions, to steer the system to operate at a high heat transfer coefficient, as shown in Fig. 2. Starting from the bottom, we find the driver layer with code to access the required hardware. For the CPU, we find temperature sensing code, computational load sensing code, and DVFS actuator code. To have the lowest overhead possible, we used the Linux MSR kernel module to access from userspace the CPU Model-Specific Registers (MSRs) directly, achieving unrestricted control of the hardware. To avoid interference from the kernel frequency governors, we selected the Linux userspace governor, effectively disabling the kernel frequency management. For the CPU, we used Model-Specific Registers (MSRs) for sensing temperature and computational load, and to drive the DVFS actuator. To control 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 throughout the entire set of experiments.

3.2. 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. Some approaches to exploit task-level parallelism include speculative task execution, even leveraging transactional memory [23]. This strategy aims at removing the need to dynamically maintain a task dependency graph by speculatively executing tasks and aborting them if dependencies are not met [24]. However, abort decisions might be based on conservative ordering constraints, possibly limiting parallelism. Another strategy is to use satisfaction games [25] to decide which tasks to offload to which device. However, those are more focused on dynamic IoT environments that are far more constrained than high-performance systems. Also, dataflow-based alternatives [26,27] aim at exploiting task-level parallelism by creating a dataflow hardware design. However, since the dataflow task graph is computed at compile-time, it may limit the level of parallelism that a dynamic approach could detect at runtime.

To solve these issues, BSC, in collaboration with UPC, developed a HW Intellectual Property (IP) core that takes care of scheduling tasks into different cores or accelerators. This serves two purposes. On the one hand, management tasks such as keeping track of the accelerator state (idle, busy, etc.), are offloaded to an external IP, freeing host CPU resources. On the other hand, it reduces the amount of communication and synchronization points between the different processing elements in a given system [28]. The scheduler IP, instead of the host processor, keeps track of released dependencies and launches tasks as their data becomes available. 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. as shown in Fig. 4(a).

In this figure, accelerators, represented as rockets, are controlled by the *HW sched* while the host processors, represented as gears, do not directly control task execution in 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, allowing

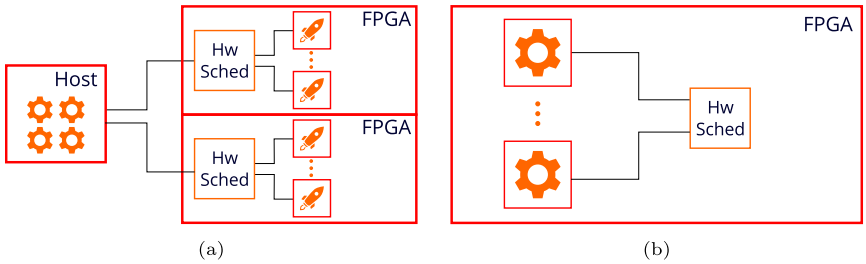


Fig. 4. Hardware scheduler diagram used to manage FPGA accelerators (a) and integrated into a RISC-V SMP system (b).

much higher throughput than the maximum achievable when the CPU directly manages accelerator execution. Moreover, by reducing the time that accelerators remain idle waiting for synchronization and scheduling, they consume static power at all times, which represents a significant amount of power in FPGA designs.

We also have integrated this IP in a Rocket Chip-based RISC-V multicore system [29]. This is shown in Fig. 4(b). In this case, the HW task scheduler is tightly coupled with the cores. More precisely, a new set of CPU instructions is added so that CPU cores can access task scheduling HW, allowing efficient task scheduling for multi-core 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 submitted to the HW scheduler to be scheduled based on their data dependencies automatically by the HW module. Then, processors request tasks to the *Hw sched*, which dispatches tasks that are ready to execute. When the execution is finished, the scheduler is notified so that data dependencies are freed and execution can progress.

3.3. 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 [30]. This results in an overall improvement in performance and energy efficiency [31]. 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) [32]. 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. Integrating the communication infrastructure in the FPGA design as opposed to perform the communications through the host system allows us to eliminate the need of moving data between host and FPGA which has a big impact both in performance and energy usage since cost of moving data from FPGA memory to main memory and back is removed. To make use of the communication infrastructure we developed a simple API to implement communications inside the accelerators. Basic API calls are shown in listing 1. This API is inspired by the MPI specification. `OMPIF_Comm_rank()` and `OMPIF_Comm_size()` are used to query cluster size and current rank. The `OMPIF_Send` and `OMPIF_Recv` primitives, along destination, data and length, have two extra arguments, `numDeps` and `deps`. These functions are non-blocking as they are intrinsic tasks. Thus, the way to synchronize them is to add dependencies with other tasks, using the aforementioned `deps` arguments, or using a `taskwait` directive. The `OMPIF_Allgather` has the same semantics as `MPI_Allgather` with `MPI_IN_PLACE` [33] in the send buffer. Finally, `OMPIF_Bcast` follows the semantics of `MPI_Bcast`. These collective primitives, as opposed to `OMPIF_Send` and `OMPIF_Recv`, are executed synchronously and they also imply a global synchronization point.

Moreover, the APEIRON framework [34] 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.

3.4. IMP: Implicit message passing for FPGA

To reduce the complexity of writing distributed applications, we propose an approach that hides the message passing layer from the user while distributing the application task graph. While there are programming models that target application acceleration using FPGA devices such as TAPA [35] or TaPaSCo [36], they do not target programming clusters of FPGAs. Our IMP approach is based on a static data decomposition of the problem data and follows the owner-computes rule [37]. Therefore, the user has to give additional information about data locality on each node. Each data block has an owner, which is the rank of the node that has at all times the updated data. For each task, the user specifies the task owner, which is the rank of the node that executes the task. With this information, the runtime determines if communication is needed for each dependency with an owner, taking into account the dependence direction and task ownership. Then, for a particular node, it decides if the task has to be executed or if it will be executed by another node. For this model to work, all nodes must execute the same code regardless of the rank, because the runtime needs the information on data ownership on all tasks of the application.

Listing 2 shows three examples of the model described in this section, written in C. In the OmpSs programming model, an `in`, `out` or `inout` clause is a list of dependence regions or points, but in this example, the clause specifies a single dependence with two arguments. The first argument is the data region or point, and the second argument is the data ownership. The `own` directive specifies task ownership.

In `example1`, there are two regular tasks `task_1` and `task_2`. The `a` array is located on rank 0, while the `b` array is on rank 1. Thus, the first task is owned by rank 0 and the second one by rank 1, because they have their respective data as an output dependence. When creating the first task, there is no communication happening since both the data and task are owned by the same node. In this case, rank 0 creates a task, while rank 1 ignores it. However, in the `task_2` task, rank 0 issues a send to rank 1 with size `n`. On the other hand, rank 1 issues the matching message receive, and creates the `task_2` task. The send of rank 0 is executed after the first task, and the receive of rank 1 is executed before the second task.

In the second example (`example2`), the `broadcast` keyword in the `task_1` pragma specifies that the associated region of data is owned by all ranks. Therefore, after the task finishes execution on its corresponding owner, it issues a broadcast task to distribute data among all nodes. The rest of the ranks issue a corresponding receive. For `example2`, we also use the OMPIF number of ranks (`nr`) to decide who is the owner of each task.

If the distribution of data is regular, data distribution and task ownership can be deduced based on output dependencies. In `example3`,

```
void OMPIF_Send(const void* data, int size, int destination,
    int tag, int numDeps, uint64_t deps[]);
void OMPIF_Recv(void* data, int size, int source, int tag,
    int numDeps, uint64_t deps[]);
void OMPIF_Allgather(void* data, int size);
void OMPIF_Bcast(void* data, int size, int root);
unsigned char OMPIF_Comm_rank();
unsigned char OMPIF_Comm_size();
```

Listing 1: OMPIF API calls.

```
void example1(int* a, int* b, int n) {
    #pragma oss task inout([n]a, 0) own(0)
    task_1(a, n);
    #pragma oss task in([n]a, 0) inout([n]b, 1) own(1)
    task_2(a, b, n);
}
void example2(int* a, int* b, int n, int nb) {
    int nr = OMPIF_Comm_size();
    for (int i = 0; i < nb; ++i)
        #pragma oss task inout(a[i*n;n], broadcast) \
        inout(b[i*n]) own(i/(nb/nr))
        task_1(a + i*n, b + i*n);
}
#pragma oss task data_dist(block, a, n*n) \
    data_dist(cyclic, b, n) \
    inout(a) in(b)
void example3(int* a, int* b, int n) {
    for (int i = 0; i < n; ++i)
        #pragma oss task in(b[i*n;n]) inout(a[i*n;n])
        task_3(a + i*n, b + i*n, n);
}
```

Listing 2: Example of IMP+OmpSs code

special clause named `data_dist` is used. The first one specifies an even block distribution of the data. This means the matrix `a` with size `n*n` is distributed among all ranks with blocks of equal size. The second clause specifies a cyclic distribution with a block size of `n`. In the example, we are distributing consecutive rows to different ranks. Since there is only one output dependence on array `a`, the compiler can use this information to calculate the task owner based on the data owner. If there were more outputs with different owners, the user would have to decide which one is used. In this example, the only dependence that implies communication is the first one because it has a cyclic distribution. Another advantage of this syntax is that users do not need to use the cluster size or rank.

3.5. Hardware IPs for low-latency inter-FPGA communication

The INFN Communication IP is the main component of the APEIRON framework. It allows direct communication between tasks deployed on the same FPGA (intra-node communication) and on different FPGAs (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 [38] and ExaNet [39,40].

As shown in Fig. 5, 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 or 10G/25G Ethernet supporting UDP/IP transport layer offloading. `Link_Ctrl` blocks instead establish the logical link between nodes and guarantee reliable communication, eventually performing error detection and correction. 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 requesting the same port. Deadlock avoidance is guaranteed by the implementation of two virtual channels for each physical one. To assess the performance of the different releases developed for the Communication IP (128-bit and 256-bit internal datapath width for both AMD Alveo U200 and U280 boards), we carried out latency and bandwidth tests for the 256-bit version.

In the testbench, the Communication IP, featuring 4 intra-node ports and 2 inter-node ports, is implemented as an RTL-IP kernel connected to the global system/board clock of 200 MHz and to 4 dispatcher/aggregator couples. We report performance results for the 256-bit internal datapath version of the design.

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. 6(a), it is 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 µs for packet payload sizes up to 1kB.

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). For source and destination buffers in BRAM, in the Oneway communication, the bandwidth tends to saturate to 37.5 Gbps, in good agreement with the 40 Gbps bandwidth of the inter-node physical channel and with the communication protocol overhead. The latter is also responsible for the small difference between the bandwidth measured with 4 kB payload messages (47.7 Gbps) and the maximum theoretical raw bandwidth of 51.2 Gbps, as shown in Fig. 6(b). The

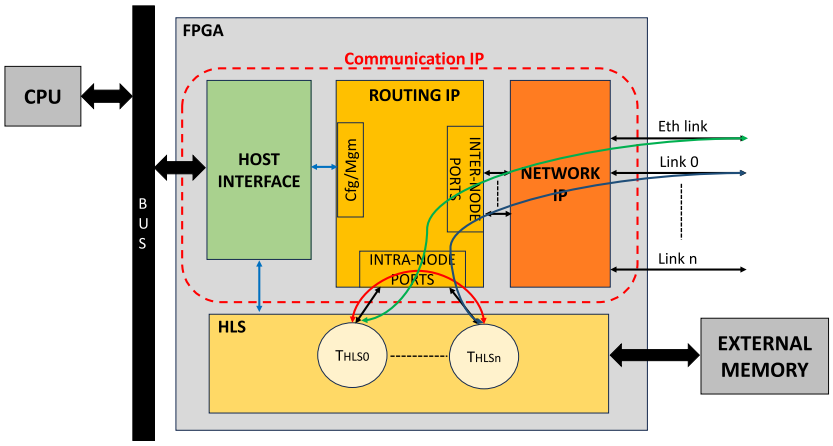
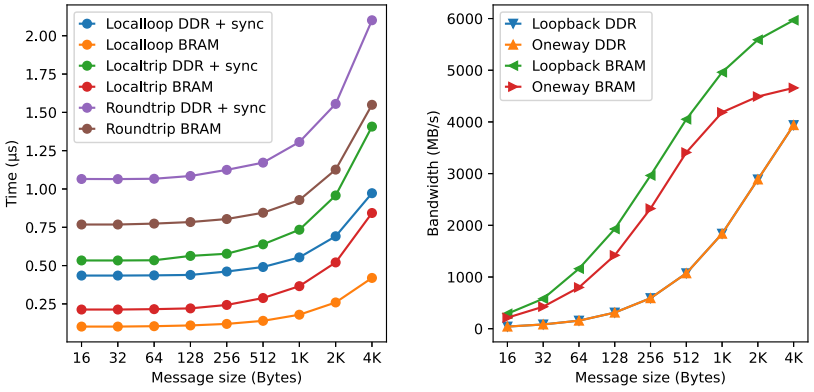


Fig. 5. Block diagram of INFN communication IP showing intra-node (red arrow) and inter-node (green and dark blue arrows) communications. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) (b)

Fig. 6. Measured latency (a) and bandwidth (b) using 256-bit internal datapath width for communication IPs.

same figure shows clearly that DDR access dominates the transfer cost. This can be seen as bandwidth is the same regardless of whether the source and destination buffers being or not in different FPGAs, as long as they are allocated in DDR memory.

3.6. 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:

3.6.1. 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 a bitstream that implements a design including the Communication IP and the HLS kernels. These HLS kernels, along with their arguments, must expose a generic AXI4-Stream interface for each communication channel as shown in listing 3.

3.6.2. 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.

A sample of the API calls is shown in listing 4 where:

```
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 3: HLS kernels prototype.

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

Listing 4: HAPECOM API pseudocode.

- `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, two APEIRON IPs manage the adaptation toward/from IntraNode ports of the INFN Communication IP: they are *Aggregator* and *Dispatcher*. These IPs are implemented as free-running stream-based HLS kernels that are integrated, along with user-specified accelerators, into the final

Vitis design. The *Aggregator* IP receives outgoing packets from the task accelerators and multiplexes them into a single stream that is then fed into the communication IP. The aggregator also builds the packet header, which is sent along the data stream to the communication IP so that the data packets can be properly routed. The *Dispatcher* IP receives incoming packets from the Routing IP, it strips the packet headers and forwards the data to the right kernel input channel.

3.6.3. 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 such as reading and writing FPGA registers or programming the FPGA. 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, which allows to perform the actions exposed by the library APIs to operate on the physical hardware. It operates on the client/server principle: it listens for user application requests over the network. At the arrival of a new client connection, a thread is generated to handle the request. This allows the server to answer multiple requests (from the same client or from different clients over the network) at the same time. The *Apeirons* 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. It allows managing the instances of the software stack distributed over the hosts in the network from a single node running the *Supervisor* module, which in the current implementation is written in Python language with a Graphical User Interface. The *Supervisor* module operates remotely on the target nodes by flashing bitstreams, running kernels, 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.

3.7. Precision tuning

Approximate computing is a wide field, revolving around the idea of trading off computation accuracy for performance and energy improvements. At the software level, approximate computing can be achieved mainly through two classes of techniques, namely perforation and precision tuning. Where perforation basically skips some operations, e.g., loop iterations, and precision tuning attempts to replace the data type employed in a particular operation with another data type for which the same operation can be performed at a lower cost – e.g., floating point operations can be replaced, under some circumstances, with fixed point operations [41]. This is effective if the floating point unit is slow, if integer units are more abundant, or if the floating unit is simply absent [42]. In the context of high-performance computing, most of the gains of precision tuning depend on the ability to replace the original data type with a smaller one, for which vectorization is possible, thus exploiting a degree of loop parallelism through compiler transformations such as strip mining [43,44].

Various tools and techniques have been explored in the literature, utilizing static analysis or code profiling to understand these trade-offs. Compiler transformations are then employed to effectively alter the precision of computations by replacing instructions and data types. Switching between different precision levels, especially when using different number system representations for real numbers (e.g., fixed point and floating point), incurs a non-negligible cost that must be considered in the trade-off [41].

The Textarossa project chooses precision tuning as a key technique for performance and energy efficiency, leveraging TAFFO [45,46], a set of plugins for the LLVM compiler framework [47]. In particular, Textarossa proposes three key extensions to TAFFO.

Support for GPGPU architectures. This extension enables the efficient use of heterogeneous computing platforms, ensuring coherent compilation of both host- and device-side code. Experimental evidence shows that misaligned data types across heterogeneous platforms can lead to significant overhead due to unnecessary type conversions, as the optimal types for each architecture may differ when considered in isolation [48]. More recently, we have been able to prove how a memory transfer-aware approach to precision tuning can bring significant improvements in memory-bound heterogeneous applications [49].

Support for high-level synthesis (HLS) tools. Integrating HLS tools, despite challenges with VitisHLS and LLVM version compatibility, allows for more flexible and efficient hardware design. VitisHLS, which forms the backbone of the Textarossa reconfigurable computing toolchain, is based on an older version of LLVM (version 8), necessitating the backporting of TAFFO to this earlier version. However, the toolchain obtained demonstrates the feasibility of integration between TAFFO and HLS systems. On the other hand, in sight of the need for long-term maintenance, and of the need of TAFFO to remain in line with modern LLVM developments (in particular the adoption of the new pass manager and of opaque pointers, both of which significantly impact TAFFO and have required a redesign of its key components), we do not consider VitisHLS integration as sustainable in the long term. Instead, we consider as a more effective approach to adopt an HLS system for which support for recent versions of LLVM was available, e.g., Bambu [50]. An alternative approach is to use TAFFO to drive the design space exploration in a platform-based scenario where a customizable floating point unit is provided [51], rather than employing HLS to implement the entire system.

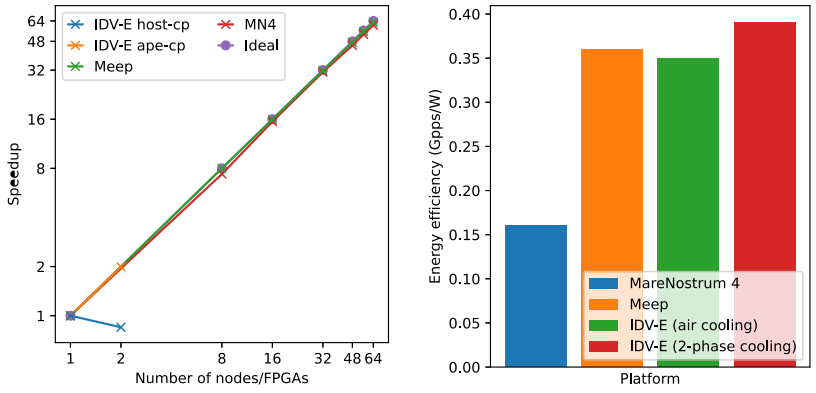
Support for the posit number system. Posits [52] offer variable precision, enabling more accurate representations at lower bit sizes compared to traditional floating-point representations. This extension broadens TAFFO's support for number systems beyond floating point and fixed point.

These advancements emphasize the potential of precision tuning in optimizing performance across various computing platforms and number systems. At the same time, integrating the TAFFO system enables the use of additional functionalities, not specifically developed within Textarossa but potentially useful, such as profile-based precision tuning, which was shown to be useful to support scenarios with high data range variability [53]. This can be the case for several long-running high-performance applications that demonstrate workload phases, a scenario that can be handled through continuous optimization techniques [54,55]. Furthermore, the technologies developed in Textarossa have been proven effective not only in the context of High Performance Computing, but also in embedded systems [56,57].

4. Evaluation and achievements

4.1. 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. For all FPGA systems, power is measured using the CMS subsystem [58], which queries FPGA board power delivery infrastructure components to obtain the total power used by the FPGA. This includes the FPGA,



(a) (b)

Fig. 7. Nobdy scalability (a) and power efficiency (b) across multiple platforms.

off-chip memory (HBM), QSFP+ interfaces, and other peripherals. CPU power is not measured as the CPUs in the cluster do not contribute to task execution and, theoretically, the FPGAs could be deployed without being attached to the host CPU [59]. For MareNostrum 4, power is reported by the task management system (Slurm). Fig. 7(a) shows the scalability of the n-body application in different scenarios.

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. 7(a), 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. 7(a), which is very close to the ideal scalability even for 64 FPGAs and a total of 576 accelerators. All in all, we can reach 74.7 Gpps (billions of particle pairs processed per second) in the IDV-E platform using 2 FPGAs (37 Gpps in a single FPGA), and 2322.98 Gpps in the larger meep platform using 64FPGAs. This comes close to the 2886.33 Gpps achieved in MareNostrum 4 [60] using 64 nodes containing a total of 128 CPUs. Other N-body accelerator architectures for HPC, such as the ones proposed by Sano et al. [61] or Del Sozzo et al. [62] reach 10.94 Gpps with an Altera Arria 10 and 13.4 Gpps using an AMD VU9P, respectively.

However, power efficiency, shown in Fig. 7(b), shows that both FPGA platforms are much more energy efficient than the CPU-based cluster. More precisely, IDV-E is 2.44 times more energy efficient than MareNostrum 4. Moreover, the IDV-E using liquid cooling developed in this project is 11% more energy efficient than the air-cooled version. Compared to the Meep platform based on FPGA on a larger scale, energy efficiency is improved by 8%, which, for a large-scale system, implies significant energy savings for this particular use case. Also, compared with a GPU of the same fabrication technology (Nvidia Geforce GTX 1060), we provide an improvement of 15% in energy efficiency [63] for the n-body application. Improvements in energy efficiency for the 2-phase cooled system is caused by the FPGA chips running at a lower temperature than the air-cooled systems, more precisely, FPGAs in the 2-phase IDV-E run at 58°C on average during application execution while the air-cooled version runs at 75°C on average. Since energy consumption increases with temperature [64,65]. This highlights that cooling technology has a significant effect on power efficiency. Moreover, running at a lower temperature can impact device durability [66] and reliability [67] this is particularly relevant since FPGAs in the air-cooled system reached a peak temperature of 90°C, while the 2-phase cooled FPGAs only reached a maximum of 60°C.

4.2. 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 FPGAs with the HLS4ML [68] tool. The CNN input data is a 16 × 16 image for each RICH physics event from its hit photomultipliers (PMTs) map, 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 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 (or from detector readout) 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 (16 × 16 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 in Host memory via an HLS kernel (krnl_receiver). This node is also in charge of measuring processing time, from the first packet sent to the last received. Therefore, data transfer time is taken into account in performance evaluation, but reading or writing from/to storage is not.
- *Computing node*: images received from external nodes are used 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. 8 shows how these components are laid out across multiple nodes and their interconnection paths between them.

RAIDER clustering quality, measured as recall and precision, is reported in Table 1 They were measured by taking 2.7M events, extracted from the NA62 database, as neural network input. To test the peak

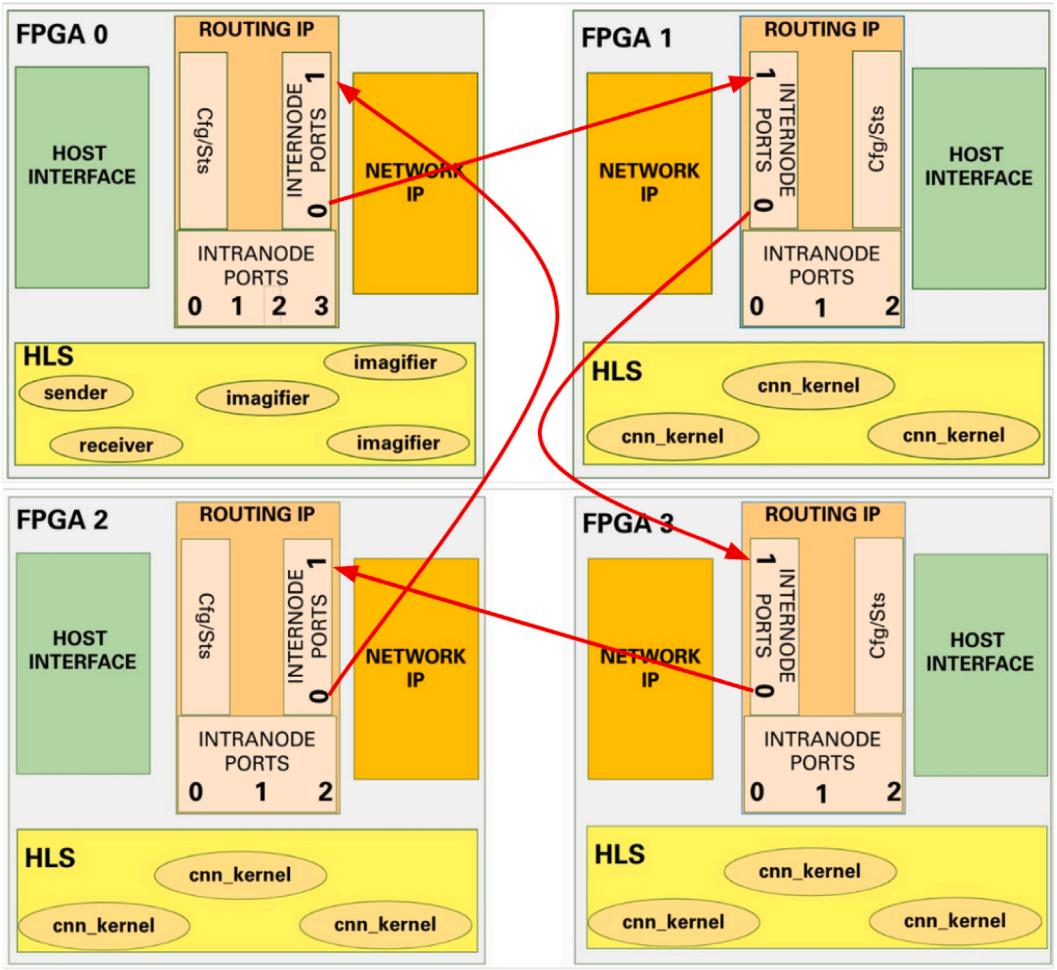


Fig. 8. Test setup of 4 AMD Alveo U280 boards installed on IDV-E nodes.

Table 1
Recall and precision values obtained from the FPGA-implemented CNN model trained on NA62 physical events number of rings classification.

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

throughput and energy efficiency values of the application, we decided to work with a subset of events sent through the network multiple times. These events are loaded from a 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 presents an initiation interval of ~160 clock cycles while loading events from DDR memory, reaching a maximum throughput of 1.278 MHz.

By doing this, we can evaluate peak processing performance, also, this scenario is closer to the target setup where events are received at a rate of 10M events per second from a dedicated interface. We run this application on two different setups: four interconnected AMD Alveo U280 installed on the IDV-E platform 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. 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, defined as the number of reconstructed events per second, we tested the system scaling from 2 (one preprocessing and one computing) FPGAs up to 4 (adding 2 more computing FPGAs). However, since the number of resources available on the Alveo U280s is larger than the Alveo U200 FPGAs of the previous testbed, we implemented 3 CNNs HLS kernels on each computing FPGA.

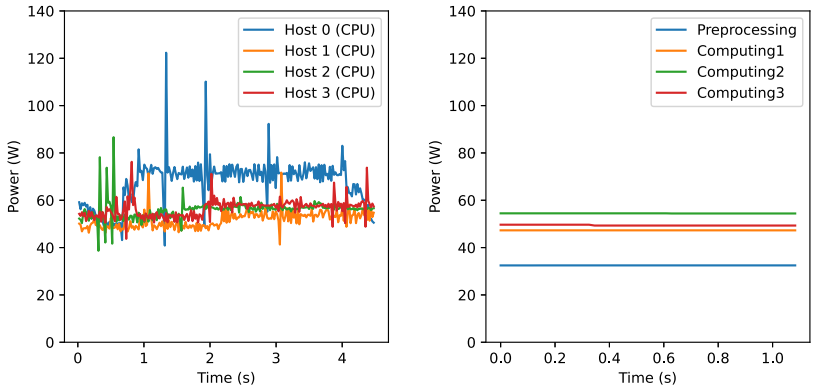
In terms of energy efficiency of the heterogeneous platform, power consumption measurements for the CPU hosts were conducted using *turbostat*, a Linux command-line tool designed to monitor processor topology, operating frequencies, idle power-state data, temperature, and power usage for X86-based systems. These measurements were collected simultaneously with the execution of the application on each CPU host in the experimental setup, with the corresponding results illustrated in Fig. 9(a).

Power consumption measurements for the FPGA were derived from the Xilinx Runtime (XRT) summary *.csv* file generated by the CPU host application. Based on the scaling of nodes and the number of CNNs, power profiles were created for each RAIDER configuration evaluated, with an example provided in Fig. 9(b).

Integrating the power consumption profiles of CPU and FPGAs, we have been able to assess the platforms' energy efficiency, defined as the number of reconstructed events per Joule.

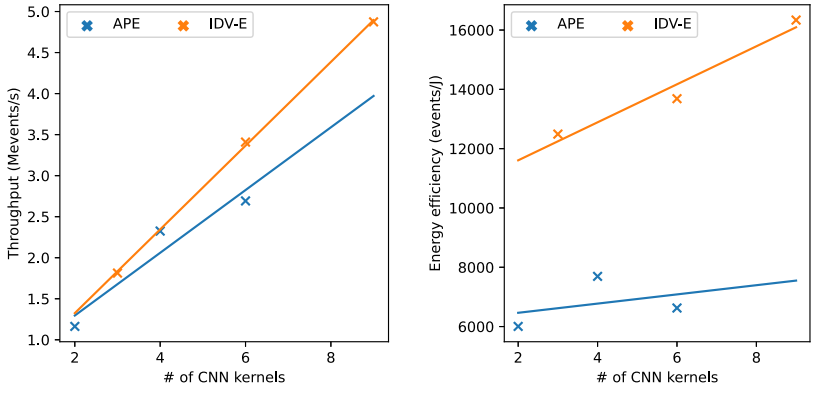
Both throughput and energy efficiency figures are presented in Table 2, scaling the number of CNN kernels.

To compare and visualize the trends of the throughput values reported in Table 2 under *APE* and *IDV-E* for the APEIRON setup and the IDV-E setups respectively, we choose to picture them in Fig. 10. From the linear regression curve obtained from the different sets of



(a) (b)

Fig. 9. RAIDER power profiles for CPU hosts (4 Intel Sapphire Rapid) (a) and FPGA (4 AMD Alveo U200 setup) (b).



(a) (b)

Fig. 10. RAIDER throughput scaling (a) and energy efficiency (b) trends for the IDV-E Alveo U280 setup (orange) and Alveo U200 (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Processing throughput and energy efficiency with an increasing number of CNN HLS kernels on AMD Alveo 200 APE Lab testbed and IDV-E Alveo U280.

APE (AMD U200)			IDV-E (AMD U280)		
#CNN	Throughput (Mevents/s)	En. Eff. (kevents/J)	#CNN	Throughput (Mevents/s)	En. Eff. (kevents/J)
2	1.163	6.005	3	1.813	12.490
4	2.325	7.692	6	3.409	13.685
6	2.692	6.626	9	4.874	16.336

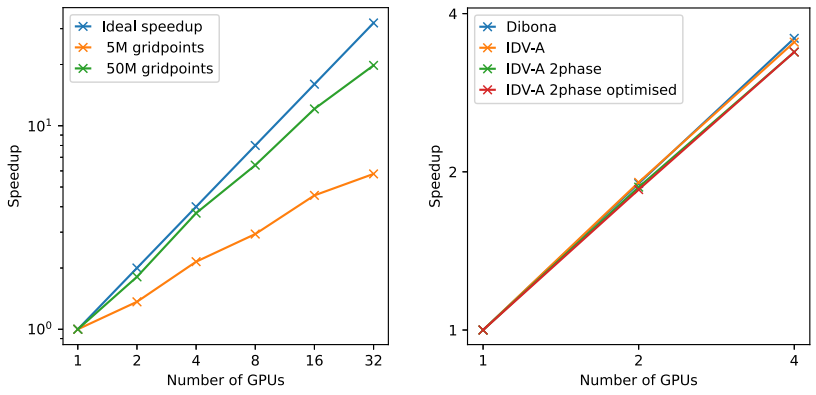
Table 3
Comparison of different CPU and GPU libraries using different precision settings for Quantum TEA. The data is for bond dimension $\chi = 64$. Simulation time and error are shown.

Setup	ZZZZ	DDDD	SSSD
F90 (CPU)	36 s, $2.5 \cdot 10^{-7}$	13 s, $1.2 \cdot 10^{-6}$	9 s, $1.4 \cdot 10^{-6}$
numpy (CPU)	272 s, $1.7 \cdot 10^{-9}$	117 s, $1.8 \cdot 10^{-9}$	39 s, $1.8 \cdot 10^{-9}$
cupy (GPU)	91 s, $1.7 \cdot 10^{-9}$	81 s, $1.7 \cdot 10^{-9}$	56 s, $1.7 \cdot 10^{-9}$
torch (CPU)	62 s, $1.7 \cdot 10^{-9}$	23 s, $1.8 \cdot 10^{-9}$	19 s, $7.3 \cdot 10^{-9}$
torch (GPU)	13 s, $1.8 \cdot 10^{-9}$	12 s, $2.0 \cdot 10^{-9}$	14 s, $2.0 \cdot 10^{-9}$

results, we can notice that the RAIDER application scales 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.

4.3. Simulating quantum systems on IDV-A

Quantum TEA is among the applications studied within Textarossa. 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 as shown in Table 3: In particular, from the numpy (CPU) ZZZZ pattern, which is the only one available at the project start, we can reach a speedup of 2.32× when using the DDDD pattern and 6.97× when using the SSSD sweep pattern. At this moderate bond dimension, one can already see a benefit of GPUs for complex arithmetic of 3× for switching from the fastest CPU code to the best GPU while decreasing the error by two orders of magnitude (qgreentea-fortran → qtealeaves-torch-gpu). These improvements have an actual impact on how this type of simulation is approached in the future.



(a) (b)

Fig. 11. UrbanAir speedup comparison for different domain sizes (a) and for different system configurations for 59M grid points (b).

4.4. UrbanAir HPC application

UrbanAir is a multiscale HPC application to predict air quality in urban environments. It is based on WRF, mesoscale community weather prediction model, and EULAG, an all-scale geophysical flow solver. Such coupling allows for solving different environmental-related problems, such as detailed weather forecasts for urban areas, detailed predictions for renewable energy sources, or assessing air quality at street level, as in this case. Detailed prediction at city or street level requires running over a domain with 10 m 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, which prevents solving 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 UrbanAir-gcrk starts with solver initialization, followed by a prediction of initial wind velocity and pressure, and initialization of boundary conditions. Eventually, the iterative solver is started, where, for each iteration, reduction and preconditioner (to speed up the solver) subroutines are called, and the Laplacian operator is solved. Every sub-routine is provided with a separate implementation for CPUs and GPUs, so that within a compilation, the user can decide whether to run on CPUs or GPUs or use a mix of them. UrbanAir-gcrk iterates over the whole domain, doing stencil computations. Every subdomain is assigned to exactly one GPU, and after each iteration, the data between subdomains is exchanged using additional HALO cells (borders of the subdomain). To increase performance on NVIDIA V100 and H100 GPU cards, and to support wind-flow-specific settings, additional improvements were provided: (i) additional boundary conditions /data were placed in shared memory to speed up, (ii) further optimization of the Laplacian operator implementation for GPUs was introduced, (iii) the number of communications in sub-kernels was limited, (iv) further optimal size of the block of threads within the subdomain for each sub-kernel is selected based on some auto-tuning. After each iteration, a global sum of variables is calculated. The reduction algorithm was improved to benefit from shared memory in a multi-GPU environment. 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, the GPowerU project tool was used, also available on the IDV-A TEXTAROSSA platform. For comparison, the tests were conducted on the PSNC Altair HPC machine equipped with NVIDIA V100 and the newly available IDV-A equipped with NVIDIA H100.

The multi-GPU version scales very well unless the problem size is not large enough, as depicted in Fig. 11(a).

Fig. 11(b) compares speedup of UrbanAir-gcrk across different platforms, configurations, and code optimizations. Dibona is the former IDV-A platform. On final IDV-A, three different configurations are compared: IDV-A with a traditional cooling system (wo-2phase), IDV-A with the two-phase cooling system installed (w-2phase), IDV-A with the two-phase cooling system on which an optimized version of UrbanAir-gcrk was run (w-phase-opt). The speedup characteristic is much the same for each configuration. The w-phase-opt achieves the best iterations/s KPI, see Fig. 12(b). It is worth mentioning that the thermal controller of the 2-phase cooling is actively monitoring GPUs for their load to ramp up their frequency whenever required. Therefore, it provides some overhead, or rather, a very slight performance decrease of applications running on GPUs. In the case of UrbanAir-gcrk, it impacts time-to-solution the more GPUs are used. Provided optimizations mitigate this impact. Compared to the initial version of IDV-A, Dibona, a 10% increase in the number of iterations per second is observed.

Fig. 12(a) 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 platform, while on the final IDV-A, three different configurations are compared: IDV-A with a traditional cooling system (*wo-2phase*), IDV-A with the two-phase cooling system installed (*w-2phase*), IDV-A with the two-phase cooling system on which an 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. 12(b)) and energy efficiency. Both are improved with the introduced optimizations.

4.5. Math library

Several modules of a mathematical library designed for GPU-accelerated heterogeneous architectures, such as IDV-A, were developed and tested throughout the project. These modules include computational kernels essential for sparse matrix computations and iterative linear solvers, which are widely used in High-Performance Computing (HPC) and High-Performance Data Analytics/Artificial Intelligence (HPDA/AI) domains. As is well known, these modules are typically bound by memory and communication limitations. The library was primarily intended as a medium-level component of the Textarossa hardware/software architecture and also served as a benchmark tool for IDV-A. Significant effort was dedicated to optimizing GPU kernel efficiency and ensuring scalability across multiple GPUs. Multi-GPU configurations are often necessary because the problem dimensions exceed the memory capacity of a single GPU. Therefore, on IDV-A,

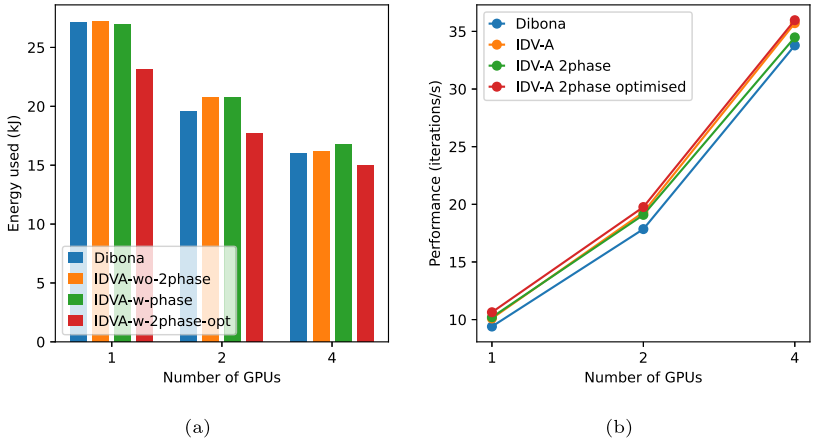


Fig. 12. UrbanAir-gcrk energy consumption (a) and performance (iterations/s) (b) for 59M grid points.

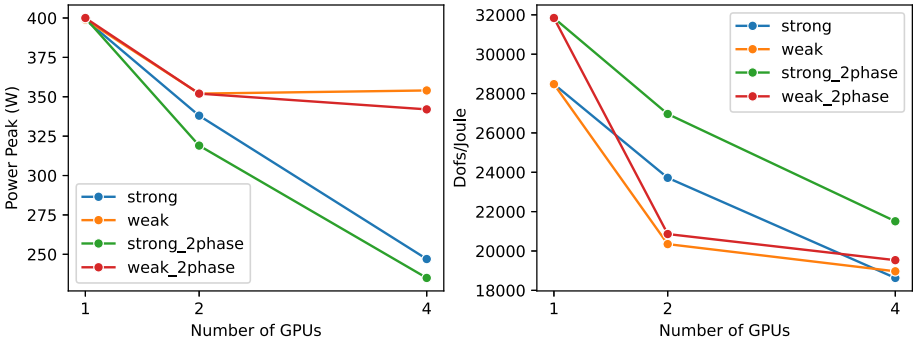


Fig. 13. Math Library power peaks and energy efficiency result reached by the solver on IDV-A before and after the installation of the new 2-phase cooling system, both in a strong and weak scaling scenario.

the library was designed to push the limits of GPU operations and memory/communication channel bandwidth at the node level. The development toolchain for the library includes C compilers, the CUDA Toolkit, and the MPI library, which are part of the basic environment of the Textarossa platform. These tools enable the reuse of highly efficient GPU kernels, originally developed for single NVIDIA GPUs, and focus on algorithms that allow scalability across large numbers of GPUs. Additionally, the GPowerU project tool, developed by INFN, was extensively used to monitor the energy consumption of GPU kernels. A dynamic measurement approach was adopted to measure energy consumption during kernel execution, excluding energy usage when GPUs are idle. The algorithms and parallel design patterns implemented for the library's kernels are thoroughly discussed in [69]. The library's benefits have been demonstrated both on IDV-A and the Italian Leonardo [70] supercomputer, with performance comparisons to the state-of-the-art NVIDIA AmgX library. For the performance and energy consumption tests on IDV-A, benchmark datasets included matrices and right-hand sides of algebraic systems arising from the Poisson equation in 3D with homogeneous Dirichlet boundary conditions and a right-hand side equal to the unit vector. This test case is a standard benchmark for sparse matrix computations, as it represents the computational kernel of many scientific and engineering applications, and is also used in the High Performance Conjugate Gradients (HPCG) benchmark. Fig. 13 presents the test results conducted on IDV-A. The main solver achieves a performance of 20.37×10^6 degrees of freedom per second (dofs/s) using four GPUs for a problem size of 244 million dofs. The corresponding energy efficiency is 19.5×10^4 dofs per Joule (dofs/J) under a weak scaling scenario. Energy consumption results from the library modules, before and after the installation of the new two-phase cooling system, indicate that the new cooling system has a minimal impact on performance. However, it appears to reduce both peak power and total energy consumption in both strong and weak scaling scenarios.

4.6. Scalability of the thermal modeling

A crucial aspect to consider when targeting HPC center-scale systems is the ability to scale up thermal and power analysis across a high number of cores within acceptable simulation times. To this end, two simulation models have been developed and integrated into the DC-WORMS power simulation tool, developed by PSNC. The first model is a general and high-accuracy transient thermal model of the evaporative cooling loop, implemented using Equation-Based Object-Oriented Modeling (EB-OOM). It is capable of performing co-simulation with the 3D-ICE chip-level thermal simulator. The chip model is coupled with the EB-OOM model representing the evaporator and the coolant cycle. The fine grain of the spatial discretization of the chip, evaporator plate, and coolant flow, as described in Section 3.1, allows for a very precise tuning of the design parameters at the cost of computational complexity. In terms of computational performance, simulating 10 s of system behavior requires over three hours of runtime, making this model well-suited for the design and optimization of the cooling system, but not ideal for large-scale thermal performance analysis at the data center level. To overcome this limitation, an optimized thermal model has been developed to balance accuracy and simulation speed. It is specifically tailored for large-scale simulations where chip floorplans or detailed power traces are not available. While the model still relies

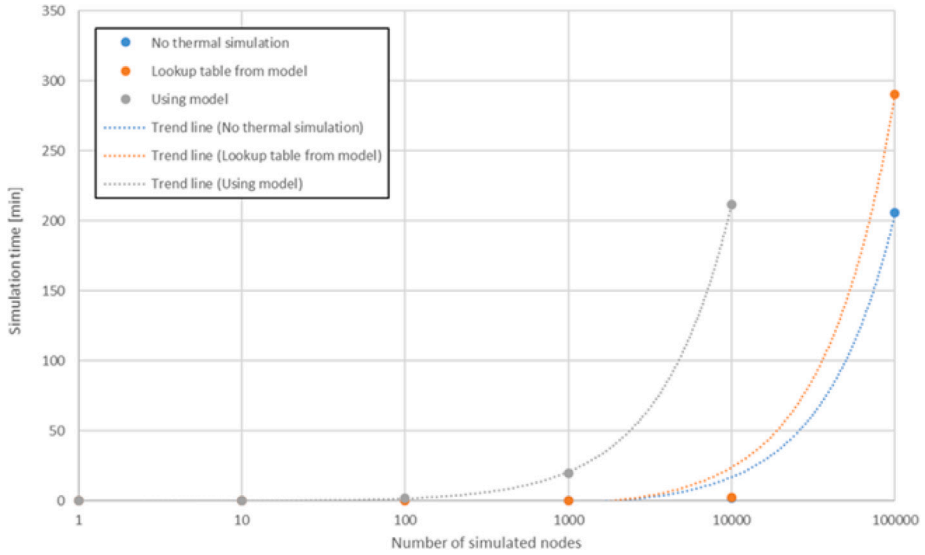


Fig. 14. DC-WORMS power and thermal simulation varying the number of cores.

on physics-inspired empirical correlations, certain simplifications have been introduced to reduce the number of equations required. The improved simulation speed enables DC-WORMS to simulate a real-scale data center. For example, simulating 10 s of operation can be completed in 30 s, with a maximum error of 6°C and a mean error of 0.75°C, representing a 300× improvement in performance. A set of experiments was conducted to assess the scalability of the DC-WORMS simulator when using the evaporative cooling model. As shown in Fig. 14, the results illustrate the simulation time as a function of the number of simulated compute nodes, as well as the impact of the evaporative cooling model on thermal simulation duration. The experiment demonstrated that simulations at this level of complexity can be executed within a reasonable time frame. Therefore, it is reasonable to expect that even more complex scenarios could be addressed in the future. Such cases may include node reliability analysis based on processor temperatures or the effect of node placement on the external loop water temperature—scenarios that were outside the scope of the TEXTAROSSA project.

5. 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 achieving high energy efficiency. Even in low-power platforms (IDV-E), evaporative cooling has been proven to provide increased energy efficiency over more classical alternatives.

The project has not only been successful in its hardware developments but also in the tools section. The IPs developed (FTS, OMPIF, APEIRON) in the project, along with the runtime (hardware - FTS - and software - Nanos6) and the programming models (OmpSs@FPGA, IMP) used have proven to be able to scale nearly ideally in systems with a high number of FPGAs (akin to IDV-E) allowing them to outperform supercomputers with the same number of nodes. Furthermore, the developed tools allow for high-level programming of such systems, which, to the best of our knowledge, is a world first. This allows users to use a familiar set of languages (C/C++) instead of HDL languages or hardware design tools to program those systems, increasing developer productivity. Also, this overcomes some of the bigger challenges when programming large-scale FPGA systems that is the lack of mature tools and ecosystems, both in the programming and design tools and the communications stack as well as the management and monitoring software. A set of additional tools for programming (TAFFO) and communicating (APEIRON) the test vehicles has been developed, providing best-in-class results. Together, these tools prove that for some applications, heterogeneous systems with FPGAs (i.e., akin to IDV-E) can provide first-level performance supercomputing with high energy and power savings.

Finally, the project has provided outstanding results in improving applications that use the test vehicles, delivering state-of-the-art performance results. Indeed, the applications presented in this paper (N-Body, RAIDER, Quantum TEA, UrbanAir, and the Math Library) are all competitive in their respective fields, delivering previously unachieved performance over the target platforms. This has been possible because the interaction between the different partners has allowed us to mix several different technologies to achieve the best results in each case. N-Body, a classical dynamical problem, has been executed in multiple FPGAs using APEIRON communication, the OmpSs@FPGA framework, the FTS IP, and the IMP programming model. RAIDER, a low-latency compliant convolutional neural network, has used a similar approach with the APEIRON framework, with the addition of using mixed precision. Quantum TEA has used the GPU test vehicle (IDV-A) and mixed precision as well. UrbanAir and the Math Library have also targeted IDV-A with the two-phase cooling, using for their improvement the power measurement tools developed in the project.

As explained in this article, the lessons learned have been multiple, and it is difficult to select the best one. On the technical side, it would probably be the necessity of adapting the solutions to the problem and not the other way around. At the same time, these solutions should be implemented in such a way that specialists in a different field could use them without the minimum hassle; otherwise, they will be lost. The last conclusion, however, is the necessity of different teams from different fields to work together, understanding each other's problems in order to reach a solution that is more than the sum of its parts.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the (EuroHPC) TEXTAROSSA project G.A. n.956831, by the Spanish Government (Grants PCI2021-121964 - TEXTAROSSA, and CEX2021-001148-S), the research project ST4HPC (PID2023-147979NB-C21) financed by MICIU/AEI /10.13039/501100011033 and by FEDER, EU, by Generalitat de Catalunya (2021 SGR 01007), and by the Barcelona Zettascale Laboratory, with the support of the Ministry for Digital Transformation and the Civil Service, within the framework of the Recovery, Transformation and Resilience Plan - Funded by the European Union - NextGenerationEU. 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.

Data availability

Data will be made available on request.

References

[1] G. Agosta, W. Fornaciari, D. Atienza, R. Canal, A. Cilardo, J. Flich Cardo, C. Hernandez Luz, M. Kulczewski, G. Massari, R. Tornero Gavilá, M. Zapater, The recipe approach to challenges in deeply heterogeneous high performance systems, Microprocess. Microsyst. 77 (2020) 103185, http://dx.doi.org/10.1016/j.micpro.2020.103185.

[2] D. Theodoropoulos, S. Mazumdar, E. Ayguade, N. Bettin, J. Bueno, S. Ermini, A. Filgueras, D. Jiménez-González, C. Álvarez Martínez, X. Martorell, F. Montefoschi, D. Oro, D. Pnevmatikatos, A. Rizzo, P. Gai, S. Garzarella, B. Morelli, A. Pomella, R. Giorgi, The axiom platform for next-generation cyber physical systems, Microprocess. Microsyst. 52 (2017) 540–555, http://dx.doi.org/10.1016/j.micpro.2017.05.018, https://www.sciencedirect.com/science/article/pii/S0141933116304434.

[3] A. Cristal, O.S. Unsal, X. Martorell, P. Carpenter, R. De La Cruz, L. Bautista, D. Jimenez, C. Alvarez, B. Salami, S. Madonar, M. Pericàs, P. Trancoso, M. vor dem Berge, G. Billung-Meyer, S. Krupop, W. Christmann, F. Klawonn, A. Mihklafi, T. Becker, G. Gaydadjiev, H. Salomonsson, D. Dubhashi, O. Port, Y. Etsion, V. Nowack, C. Fetzer, J. Hagemeyer, T. Jungeblut, N. Kucza, M. Kaiser, M. Porrmann, M. Pasin, V. Schiavoni, I. Rocha, C. Göttel, P. Felber, Legato: towards energy-efficient, secure, fault-tolerant toolset for heterogeneous computing, in: Proceedings of the 15th ACM International Conference on Computing Frontiers, CF '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 276–278, http://dx.doi.org/10.1145/3203217.3205339.

[4] Co-designed innovation and system for resilient exascale computing in Europe: from applications to silicon, 2021, http://dx.doi.org/10.3030/754337.

[5] European joint effort toward a highly productive programming environment for heterogeneous exascale computing (epeec), 2022, http://dx.doi.org/10.3030/801051.

[6] O. Aumage, V. Bartsch, G. Beckett, M. Bull, Intertwine, programming model interoperability towards exascale, h2020, Impact 2018 (5) (2018) 45–47, http://dx.doi.org/10.21820/23987073.2018.5.45, https://www.ingentaconnect.com/content/sil/impact/2018/00002018/00000005/art00015.

[7] L. Papadopoulos, D. Soudris, C. Kessler, A. Ernstsson, J. Ahlqvist, N. Vasilas, A.I. Papadopoulos, P. Seferlis, C. Prouveur, M. Haefele, S. Thibault, A. Salamanis, T. Ioakimidis, D. Kehagias, Exa2pro: A framework for high development productivity on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 33 (4) (2022) 792–804, http://dx.doi.org/10.1109/TPDS.2021.3104257.

[8] A. Rigo, C. Pinto, K. Pouget, D. Raho, D. Dutoit, P.Y. Martinez, C. Doran, L. Benini, I. Mavroidis, M. Marazakis, V. Bartsch, G. Lonsdale, A. Pop, J. Goodacre, A. Colliot, P. Carpenter, P. Radojković, D. Pleiter, D. Drouin, B. Dupont de Dinechin, Paving the way towards a highly energy-efficient and highly integrated compute node for the exascale revolution: The exanode approach, in: 2017 Euromicro Conference on Digital System Design, DSD, 2017, pp. 486–493, http://dx.doi.org/10.1109/DSD.2017.37.

[9] M. Katevenis, R. Ammendola, A. Biagioni, P. Cretaro, O. Frezza, F. Lo Cicero, A. Lonardo, M. Martinelli, P.S. Paolucci, E. Pastorelli, F. Simula, P. Vicini, G. Taffoni, J.A. Pascual, J. Navaridas, M. Luján, J. Goodacre, B. Lietzow, A. Mouzakitis, N. Chrysos, M. Marazakis, P. Gorlani, S. Cozzini, G.P. Brandino, P. Koutsourakis, J. van Ruth, Y. Zhang, M. Kersten, Next generation of exascale-class systems: Exanest project and the status of its interconnect and storage development, Microprocess. Microsyst. 61 (2018) 58–71, http://dx.doi.org/10.1016/j.micpro.2018.05.009, https://www.sciencedirect.com/science/article/pii/S0141933118300188.

[10] J. Flich, G. Agosta, P. Ampletzer, D.A. Alonso, C. Brandolese, E. Cappe, A. Cilardo, L. Dragić, A. Dray, A. Duspara, W. Fornaciari, E. Fusella, M. Gagliardi, G. Guillaume, D. Hofman, Y. Hoornenborg, A. Iranfar, M. Kovač, S. Libutti, B. Maitre, J.M. Martínez, G. Massari, K. Meinds, H. Mlinarić, E. Papastefanakis, T. Picornell, I. Piljić, A. Pupykina, F. Reghenzani, I. Staub, R. Tornero, M. Zanella, M. Zapater, D. Zoni, Exploring manycore architectures for next-generation hpc systems through the mango approach, Microprocess. Microsyst. 61 (2018) 154–170, http://dx.doi.org/10.1016/j.micpro.2018.05.011, https://www.sciencedirect.com/science/article/pii/S0141933118300243.

[11] C. Silvano, G. Agosta, S. Cherubin, D. Gadioli, G. Palermo, A. Bartolini, L. Benini, J. Martinovič, M. Palkovič, K. Slaninová, J.a. Bispo, J.a.M.P. Cardoso, R. Abreu, P. Pinto, C. Cavazzoni, N. Sanna, A.R. Beccari, R. Cmar, E. Rohou, The antarex approach to autotuning and adaptivity for energy efficient hpc systems, in: Proceedings of the ACM International Conference on Computing Frontiers, CF '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 288–293, http://dx.doi.org/10.1145/2903150.2903470.

[12] J. Garcia-Blas, J. Carretero, Aspide: exascale programing models for extreme data processing, in: Proceedings of the 20th ACM International Conference on Computing Frontiers, CF '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 279–284, http://dx.doi.org/10.1145/3587135.3592173.

[13] D. Groen, H. Arabnejad, V. Jancauskas, W. Edeling, F. Jansson, R.A. Richardson, J. Lakhlili, L. van Veen, B. Bosak, P. Kopta, D.W. Wright, N. Monnier, P. Karlshoefer, D. Suleimenova, R. Sinclair, M. Vassaux, A. Nikishova, M. Bieniek, O.O. Luk, M. Kulczewski, E. Raffin, D. Crommelin, O. Hoenen, D.P. Coster, T. Piontek, P.V. Coveney, Vecmatk: A scalable verification, validation and uncertainty quantification toolkit for scientific simulations, Phil. Trans. R. Soc. A 379 (2021) http://dx.doi.org/10.1098/rsta.2020.0221.

[14] Computing patterns for high performance multiscale computing, 2018, http://dx.doi.org/10.3030/671564.

[15] G. Agosta, M. Aldinucci, C. Alvarez, R. Ammendola, Y. Arfat, O. Beaumont, M. Bernaschi, A. Biagioni, T. Boccali, B. Bramas, C. Brandolese, B. Cantalupo, M. Carrozzo, D. Cattaneo, A. Celestini, M. Celino, I. Colonnelli, P. Cretaro, P. D'Ambra, M. Danelutto, R. Esposito, L. Eyraud-Dubois, A. Filgueras, W. Fornaciari, O. Frezza, A. Galimberti, F. Giacomini, B. Goglin, D. Gregori, A. Guermouche, F. Iannone, M. Kulczewski, F. Lo Cicero, A. Lonardo, A.R. Martinelli, M. Martinelli, X. Martorell, G. Massari, S. Montangero, G. Mittone, R. Namyst, A. Oleksiak, P. Palazzari, P.S. Paolucci, F. Reghenzani, C. Rossi, S. Saponara, F. Simula, F. Terraneo, S. Thibault, M. Torquati, M. Turisini, P. Vicini, M. Vidal, D. Zoni, G. Zummo, Towards extreme scale technologies and accelerators for eurohpc hw/sw supercomputing applications for exascale: The textarossa approach, Microprocess. Microsyst. 95 (2022) 104679, http://dx.doi.org/10.1016/j.micpro.2022.104679, https://www.sciencedirect.com/science/article/pii/S0141933122002095.

[16] E. Perdomo, A. Kropotov, F.K. Cano Ladino, S. Zafar, T. Cervero, X.M. Bofill, B. Salami, 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, RAPIDO '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 29–34, http://dx.doi.org/10.1145/3642921.3642928.

[17] B.S. Center, Marenostrum supercomputer, 2024, https://bsc.es/marenostrum/marenostrum/technical-information.

[18] G. Massari, M. Peta, A. Campi, F. Reghenzani, F. Terraneo, G. Agosta, W. Fornaciari, S. Ciesielski, M. Kulczewski, W. Piatek, Reliability-oriented resource management for high-performance computing, Sustain. Comput.: Inform. Syst. 39 (2023) 100873, http://dx.doi.org/10.1016/j.suscom.2023.100873.

[19] A. Iranfar, F. Terraneo, G. Csordas, M. Zapater, W. Fornaciari, D. Atienza, Dynamic thermal management with proactive fan speed control through reinforcement learning, in: Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition, DATE 2020, 2020, pp. 418–423, http://dx.doi.org/10.23919/DATE48585.2020.9116510.

[20] W. Fornaciari, F. Terraneo, G. Agosta, Z. Giuseppe, L. Saraceno, G. Lancione, D. Gregori, M. Celino, The textarossa approach to thermal control of future hpc systems, in: Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer, Cham, 2022, pp. 420–433.

[21] F. Terraneo, A. Leva, W. Fornaciari, M. Zapater, D. Atienza, 3D-ICE 3.0: efficient nonlinear MPSoC thermal simulation with pluggable heat sink models, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2021) http://dx.doi.org/10.1109/TCAD.2021.3074613, 1–1.

[22] A. Leva, F. Terraneo, I. Giacomello, W. Fornaciari, Event-based power/performance-aware thermal management for high-density microprocessors, IEEE Trans. Control Syst. Technol. 26 (2) (2018) 535–550, http://dx.doi.org/10.1109/TCST.2017.2675841.

[23] M.C. Jeffrey, S. Subramanian, C. Yan, J. Emer, D. Sanchez, Unlocking ordered parallelism with the swarm architecture, IEEE Micro 36 (3) (2016) 105–117, http://dx.doi.org/10.1109/MM.2016.12.

[24] S. Subramanian, M.C. Jeffrey, M. Abeydeera, H.R. Lee, V.A. Ying, J. Emer, D. Sanchez, Fractal: An execution model for fine-grain nested speculative parallelism, in: Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 587–599, http://dx.doi.org/10.1145/3079856.3080218.

[25] N. Irtija, I. Anagnostopoulos, G. Zervakis, E.E. Tsiropoulou, H. Amrouch, J. Henkel, Energy efficient edge computing enabled by satisfaction games and approximate computing, IEEE Trans. Green Commun. Netw. 6 (1) (2022) 281–294, http://dx.doi.org/10.1109/TGCN.2021.3122911.

[26] Y. Chi, L. Guo, J. Lau, Y.k. Choi, J. Wang, J. Cong, Extending high-level synthesis for task-parallel programs, in: 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, 2021, pp. 204–213, http://dx.doi.org/10.1109/FCCM51124.2021.00032.

[27] G. Gobieski, S. Ghosh, M. Heule, T. Mowry, T. Nowatzki, N. Beckmann, B. Lucia, Riptide: A programmable, energy-minimal dataflow compiler and architecture, in: 2022 55th IEEE/ACM International Symposium on Microarchitecture, MICRO, 2022, pp. 546–564, http://dx.doi.org/10.1109/MICRO56248.2022.00046.

[28] J. Bosch, M. Vidal, A. Filgueras, D. Jiménez-González, C. Álvarez, X. Martorell, E. Ayguadé, Task-based programming models for heterogeneous recurrent workloads, in: S. Derrien, F. Hannig, P.C. Diniz, D. Chillet (Eds.), Applied Reconfigurable Computing. Architectures, Tools, and Applications, Springer International Publishing, Cham, 2021, pp. 108–122.

[29] L. Morais, C. Álvarez, D. Jiménez-González, J.M. de Haro, G. Araujo, M. Frank, A. Goldman, X. Martorell, Enabling hw-based task scheduling in large multicore architectures, IEEE Trans. Comput. 73 (1) (2024) 138–151, http://dx.doi.org/10.1109/TC.2023.3323781.

[30] A. Filgueras, M. Vidal, D. Jiménez-González, C. Álvarez, X. Martorell, Improving performance of hpc kernels on fpgas using high-level resource management, in: 2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, 2023, http://dx.doi.org/10.1109/FCCM57271.2023.00041, 213–213.

[31] A. Filgueras, M. Vidal, D. Jiménez-González, C. Álvarez, X. Martorell, Fpga framework improvements for hpc applications, in: 2023 International Conference on Field Programmable Technology, ICFPT, 2023, pp. 286–287, http://dx.doi.org/10.1109/ICFPT59805.2023.00048.

[32] J.M. de Haro, R. Cano, C. Álvarez, D. Jiménez-González, X. Martorell, E. Ayguadé, J. Labarta, F. Abel, B. Ringlein, B. Weiss, Ompss@cloudfpga: An fpga task-based programming model with message passing, in: IPDPS'22, 2022, pp. 828–838, http://dx.doi.org/10.1109/IPDPS53621.2022.00085.

[33] Message passing interface forum: MPI: A message-passing interface standard version 4.0, 2021, https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

[34] Roberto Ammendola, et al., Apeiron: A framework for high level programming of dataflow applications on multi-fpga systems, EPJ Web Conf. 295 (2024) 11002, http://dx.doi.org/10.1051/epjconf/202429511002.

[35] L. Guo, Y. Chi, J. Lau, L. Song, X. Tian, M. Khatti, W. Qiao, J. Wang, E. Ustun, Z. Fang, Z. Zhang, J. Cong, Tapa: A scalable task-parallel dataflow programming framework for modern fpgas with co-optimization of hls and physical design, ACM Trans. Reconfigurable Technol. Syst. 16 (4) (2023) http://dx.doi.org/10.1145/3609335.

[36] C. Heinz, J. Hofmann, J. Korinth, L. Sommer, L. Weber, A. Koch, The tapasco open-source toolflow, J. Signal Process. Syst. 93 (5) (2021) 545–563, http://dx.doi.org/10.1007/s11265-021-01640-8.

[37] J.M. de Haro Ruiz, C. Álvarez Martínez, D. Jiménez-González, X. Martorell, T. Ueno, K. Sano, B. Ringlein, F. Abel, B. Weiss, Automated parallel execution of distributed task graphs with fpga clusters, Future Gener. Comput. Syst. 160 (2024) 808–824, http://dx.doi.org/10.1016/j.future.2024.06.041, https://www.sciencedirect.com/science/article/pii/S0167739X24003418.

[38] R. Ammendola, A. Biagioni, O. Frezza, A. Lonardo, F.L. Cicero, P.S. Paolucci, D. Rossetti, F. Simula, L. Tosoratto, P. Vicini, Apenet+ 34 gbps data transmission system and custom transmission logic, J. Instrum. 8 (12) (2013) C12022, http://dx.doi.org/10.1088/1748-0221/8/12/C12022.

[39] R. Ammendola, et al., Large scale low power computing system: Status of network design in exanest and euroexa projects, Adv. Parallel Comput. 32 (2018) 750–759, http://dx.doi.org/10.3233/978-1-61499-843-3-750.

[40] M. Katevenis, R. Ammendola, et al., Next generation of exascale-class systems: Exanest project and the status of its interconnect and storage development, Microprocess. Microsyst. 61 (2018) 58–71, http://dx.doi.org/10.1016/j.micpro.2018.05.009.

[41] S. Cherubin, G. Agosta, Tools for reduced precision computation: a survey, ACM Comput. Surv. 53 (2) (2020) http://dx.doi.org/10.1145/3381039.

[42] D. Cattaneo, M. Chiari, N. Fossati, S. Cherubin, G. Agosta, Architecture-aware precision tuning with multiple number representation systems, in: 2021 58th ACM/IEEE Design Automation Conference, DAC, 2021, pp. 673–678, http://dx.doi.org/10.1109/DAC18074.2021.9586303.

[43] S. Cherubin, et al., Implications of reduced-precision computations in hpc: Performance, energy and error, in: Parallel Computing Is Everywhere, IOS Press, 2018, pp. 297–306.

[44] D.F. Bacon, S.L. Graham, O.J. Sharp, Compiler transformations for high-performance computing, ACM Comput. Surv. 26 (4) (1994) 345–420, http://dx.doi.org/10.1145/197405.197406.

[45] S. Cherubin, D. Cattaneo, M. Chiari, G. Agosta, Dynamic precision autotuning with taffo, ACM Trans. Archit. Code Optim. 17 (2) (2020) http://dx.doi.org/10.1145/3388785.

[46] D. Cattaneo, M. Chiari, G. Agosta, S. Cherubin, Taffo: The compiler-based precision tuner, SoftwareX 20 (2022) 101238.

[47] C. Lattner, V. Adve, LLVM: A compilation framework for lifelong program analysis & transformation, in: CGO'04, CGO '04, IEEE Computer Society, 2004, p. 75.

[48] D. Cattaneo, A. Maggioli, G. Magnani, L. Denisov, S. Yang, G. Agosta, S. Cherubin, Mixed precision in heterogeneous parallel computing platforms via delayed code analysis, in: International Conference on Embedded Computer Systems, Springer, 2023, pp. 469–477.

[49] G. Magnani, D. Cattaneo, L. Denisov, G. Tagliavini, G. Agosta, S. Cherubin, Synergistic memory optimisations: precision tuning in heterogeneous memory hierarchies, IEEE Trans. Comput. (2025).

[50] F. Ferrandi, V.G. Castellana, S. Curzel, P. Fezzardi, M. Fiorito, M. Lattuada, M. Minutoli, C. Pilato, A. Tumeo, Invited: Bambu: an open-source research framework for the high-level synthesis of complex applications, in: 2021 58th ACM/IEEE Design Automation Conference, DAC, 2021, pp. 1327–1330, http://dx.doi.org/10.1109/DAC18074.2021.9586110.

[51] L. Denisov, A. Galimberti, D. Cattaneo, G. Agosta, D. Zoni, Design-time methodology for optimizing mixed-precision cpu architectures on fpga, J. Syst. Archit. 155 (2024) 103257.

[52] J.L. Gustafson, I.T. Yonemoto, Beating floating point at its own game: Posit arithmetic, Supercomput. Front. Innov. 4 (2) (2017) 71–86.

[53] L. Denisov, et al., The impact of profiling vs static analysis in precision tuning, IEEE Access (2024).

[54] T. Kistler, M. Franz, Continuous program optimization: A case study, ACM Trans. Program. Lang. Syst. 25 (4) (2003) 500–548, http://dx.doi.org/10.1145/778559.778562.

[55] S. Cherubin, G. Agosta, Libversioningcompiler: An easy-to-use library for dynamic generation and invocation of multiple code versions, SoftwareX 7 (2018) 95–100.

[56] W. Fornaciari, G. Agosta, D. Cattaneo, L. Denisov, A. Galimberti, G. Magnani, D. Zoni, Hardware and software support for mixed precision computing: A roadmap for embedded and hpc systems, in: Proceedings of 2023 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2023, pp. 1–6, http://dx.doi.org/10.23919/DATE56975.2023.10137092, https://ieeexplore.ieee.org/document/10137092.

[57] C.B. Ciobanu, et al., Risc-v accelerators, enablement and applications for automotive and smart home in the isolde project, in: International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer, 2024, pp. 215–230.

[58] Advanced micro devices: alveo card management solution subsystem product guide (PG348), 2024, https://docs.amd.com/r/en-US/pg348-cms-subsystem.

[59] F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, S. Paredes, An fpga platform for hyperscalers, in: 2017 IEEE 25th Annual Symposium on High-Performance Interconnects, HOTI, 2017, pp. 29–32, http://dx.doi.org/10.1109/HOTI.2017.13.

[60] J.M. de Haro Ruiz, C. Ávarez Martínez, D. Jiménez-González, X. Martorell Bofill, Enabling high-level parallel programming on multi-fpga clusters, in: Proceedings of the 14th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, HEART '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–9, http://dx.doi.org/10.1145/3665283.3665292.

[61] K. Sano, S. Abiko, T. Ueno, Fpga-based stream computing for high-performance n-body simulation using floating-point dsp blocks, in: Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, HEART '17, Association for Computing Machinery, New York, NY, USA, 2017, http://dx.doi.org/10.1145/3120895.3120909.

[62] E. Del Sozzo, M. Rabozzi, L. Di Tucci, D. Sciuto, M.D. Santambrogio, A scalable fpga design for cloud n-body simulation, in: 2018 IEEE 29th International Conference on Application-Specific Systems, Architectures and Processors, ASAP, 2018, pp. 1–8, http://dx.doi.org/10.1109/ASAP.2018.8445106.

[63] J.M. de Haro, J. Bosch, A. Filgueras, M. Vidal, D. Jiménez-González, C. Álvarez, X. Martorell, E. Ayguadé, J. Labarta, Ompss@fpga framework for high performance fpga computing, IEEE Trans. Comput. 70 (12) (2021) 2029–2042, http://dx.doi.org/10.1109/TC.2021.3086106.

[64] P. Kocanda, A. Kos, Static and dynamic energy losses vs. temperature in different cmos technologies, in: 2015 22nd International Conference Mixed Design of Integrated Circuits & Systems, MIXDES, 2015, pp. 446–449, http://dx.doi.org/10.1109/MIXDES.2015.7208560.

[65] A. Gołda, A. Kos, Temperature influence on energy losses in mosfet capacitors, Microelectron. Reliab. 44 (7) (2004) 1115–1121, http://dx.doi.org/10.1016/j.microrel.2004.03.004, https://www.sciencedirect.com/science/article/pii/S0026271404000848.

[66] J. Srinivasan, S. Adve, P. Bose, J. Rivers, The case for lifetime reliability-aware microprocessors, in: Proceedings. 31st Annual International Symposium on Computer Architecture, 2004, pp. 276–287, http://dx.doi.org/10.1109/ISCA.2004.1310781.

[67] F. Koc, B. Salami, O. Ergin, O. Unsal, A.C. Kestelman, Can we trust undervolting in fpga-based deep learning designs at harsh conditions? IEEE Micro 42 (3) (2022) 57–65, http://dx.doi.org/10.1109/MM.2022.3153891.

[68] J. Duarte, et al., Fast inference of deep neural networks in FPGAs for particle physics, J. Instrum. 13 (07) (2018) P07027, http://dx.doi.org/10.1088/1748-0221/13/07/P07027.
[69] M. Bernaschi, A. Celestini, F. Vella, P. D'Ambra, A multi-GPU aggregation-based AMG preconditioner for iterative linear solvers, IEEE Trans. Parallel Distrib. Syst. 34 (8) (2023) 2365–2376, http://dx.doi.org/10.1109/TPDS.2023.3287238.
[70] M. Turisini, M. Cestari, G. Amati, Leonardo: A pan-european preexascalesupercomputer for hpc and ai applications, J. Large-Scale Res. Facil. (JLSRF) 9 (2024) http://dx.doi.org/10.17815/jlsrf-8-186.

Antonio Filgueras earned a degree in Computer Science from Universitat Politècnica de Catalunya (UPC) in 2012. Currently, he's a Ph.D. student in the departemt of Computer Architecture at UPC and working in the OmpSs@FPGA team within the Programming Models group at BSC, focused primarily on programming models for reconfigurable systems in High Performance Computing. He has been a researcher in the AXIOM, Legato, EuroEXA, MEEP, Textarossa European projects among others.