ControlPULP: A RISC-V On-Chip Parallel Power Controller for Many-Core HPC Processors with FPGA-Based Hardware-In-The-Loop Power and Thermal Emulation

Alessandro Ottaviano[1] · Robert Balas[1] · Giovanni Bambini[2] ·
Antonio Del Vecchio[2] · Maicol Ciani[2] · Davide Rossi[2] · Luca Benini[1,2] ·
Andrea Bartolini[2]

Abstract

High-performance computing (HPC) processors are nowadays integrated cyber-physical systems demanding complex and high-bandwidth closed-loop power and thermal control strategies. To efficiently satisfy real-time multi-input multi-output (MIMO) optimal power requirements, high-end processors integrate an on-die power controller system (PCS). While traditional PCSs are based on a simple microcontroller (MCU)-class core, more scalable and flexible PCS architectures are required to support advanced MIMO control algorithms for managing the ever-increasing number of cores, power states, and process, voltage, and temperature variability. This paper presents ControlPULP, an open-source, HW/SW RISC-V parallel PCS platform consisting of a single-core MCU with fast interrupt handling coupled with a scalable multi-core programmable cluster accelerator and a specialized DMA engine for the parallel acceleration of real-time power management policies. ControlPULP relies on FreeRTOS to schedule a reactive power control firmware (PCF) application layer. We demonstrate ControlPULP in a power management use-case targeting a next-generation 72-core HPC processor. We first show that the multi-core cluster accelerates the PCF, achieving 4.9x speedup compared to single-core execution, enabling more advanced power management algorithms within the control hyper-period at a shallow area overhead, about 0.1% the area of a modern HPC CPU die. We then assess the PCS and PCF by designing an FPGA-based, closed-loop emulation framework that leverages the heterogeneous SoCs paradigm, achieving DVFS tracking with a mean deviation within 3% the plant's thermal design power (TDP) against a software-equivalent model-in-the-loop approach. Finally, we show that the proposed PCF compares favorably with an industry-grade control algorithm under computational-intensive workloads.

Keywords RISC-V · HPC processor · Power and thermal control · Scalable · Parallel microcontroller · PULP

Extended author information available on the last page of the article

  Springer

1 Introduction

After the end of Dennard's scaling, the increase in power density has become an undesired but unavoidable collateral effect of the performance gain obtained with integrated systems' technological scaling. An increase in power density has multiple adverse effects, which are collectively referred to as the *power wall*: component's lifetime shortening, electromigration, dielectric breakdown due to thermal hot spots and sharp thermal gradients, and degraded operating speed due to leakage current exponentially increasing with temperature. This trend has made the processing elements (PEs) at the heart of computing nodes energy, power, and thermally constrained [1].

Two approaches have been adopted to mitigate the *power wall* at the system level [2]: static thermal management (STM) and dynamic thermal management (DTM) techniques. The former allows increasing the thermal design power (TPD) sustained by the chip with a tailored design of heat sinks, fans, and liquid cooling. However, STM strategies incur increasingly unsustainable costs when over-designed to remove heat in the worst-case conditions for today's high performance computing (HPC) processors. Consequently, DTM techniques have become more and more crucial to bound the operating temperature with *run-time active control*, for example, by exploiting Process, Voltage, Temperature (PVT) sensors along with dynamic voltage and frequency scaling (DVFS), thread migration/scheduling, throttling, and clock gating. Hence, standard cooling systems can be designed to handle the average case, leaving the management of peaks to active control.

Modern high-performance processors feature many cores integrated into a single silicon die. Recent notable examples are AWS Graviton 3 (64 Arm Neoverse V1 cores) [3], Intel Raptor Lake (24 cores, 32 threads) [4], AMD Epyc 7004 Genoa (up to 96 Zen 4 cores) [5], SiPearl Rhea Processor (72 Arm Neoverse V1 Zeus cores) [6], Ampere Altra Max (128 Arm Neoverse N1), and the NVIDIA Grace central processing unit (CPU) (144 Arm Neoverse V2 cores). Their application workload requires runtime dynamic trade-off between maximum performance (fastest operating point [7]) in CPU-bound execution phases and energy efficiency in memory-bound execution phases (energy-aware CPU [8]).

While software-centric advanced DTM policies have been proposed [2, 9, 10], they mainly execute on the CPU's application-class processors (APs), playing the role of high-level controllers (HLCs) governors. Nevertheless, in recent years it has become clear the trend of abstracting power and system management tasks away from the APs[1] [11] towards control systems that are closer to the controlled hardware components and can guarantee faster, and more precise control actions, namely low-level controllers (LLCs).

Modern processors integrate on-die LLCs [12] in the *uncore* domain, referred to as power controller systems (PCSs), as dedicated embedded hardware resources, co-designed with a power control firmware (PCF) implementing complex multiple-input

---

[1] https://github.com/Arm-software/SCP-firmware.

multiple-output (MIMO) power management policies. Advanced DTM involves embedding and interleaving a plurality of activities in the PCS, namely (i) dynamic control of the CPU power consumption with short time constants [13], required to prevent thermal hazards and to meet the TDP limit (power capping [14]), (ii) real-time interaction with commands provided by on-die (Operating System (OS)— power management interfaces and on-chip sensors) and off-die (Baseboard Management Controller (BMC), voltage regulator module (VRM)) units and (iii) dynamic power budget allocation between general-purpose (central processing units (CPUs)) and other integrated subsystems, such as graphic processing units (GPUs) [8].

Existing on-die power controller systems (PCSs) share a similar design structure. They feature an embedded single-core microcontroller unit (MCU) [1] supported by dedicated hardware state machines [8] or more generic accelerators [15]. The hardware typically takes advantage of specific software libraries[2] to implement the real-time execution environment required to run power management policies under tight timing constraints. Many-core power management demands fine-grained control of the operating points of the processing elements (PEs) [12] to meet a given processor power budget while minimizing performance penalties. The control policy has to provide fast and predictable responses to promptly handle the incoming requests from the OS or BMC and prevent thermal hazards. A flexible and scalable way to sustain these computationally intensive operations is required to provide accurate control per core and to support more advanced control policies, such as those based on model-predictive control [9].

Furthermore, simultaneous hardware (HW) and software (SW) development of power controller system (PCS) and power control firmware (PCF) has to be coupled with a dedicated co-design and validation framework within the *controlled system*. Indeed PCS and PCF performance directly depends on their interaction with the physical state of the controlled processors—temperature, power, workload, and control decisions from the high-level controllers (HLCs). Such a physical state time-scale model requires near real-time speed to be meaningful. This involves, on the one hand, the design of adequate on-chip interfaces between the controller to be validated—integrated system under test (ISUT)—and the surrounding system. On the other, a proper *virtual* representation of the system surrounding the ISUT that models physical components on a target computer [16], called *plant model*. In a design relying on dedicated PCSs, the controlled plant is a complex multiple-input multiple-output (MIMO) multi-processor system on chip (MPSoC) with application-class processors (APs) that is often not available during the design phase of the PCS to be integrated, thus being replaced by a thermal and power model encapsulating floorplan, power and thermal information of the central processing unit (CPU) under a particular application workload.

Today's advanced heterogeneous system on chip (HeSoC) integrate both single/ multiprocessors microcontroller units (MCUs) and configurable hardware components on the same die (field-programmable gate array (FPGA)-system on chip

---

[2] https://github.com/open-power.

(SoC)). Their native integrated and configurable structure makes them ideal options for closed-loop emulation of on-chip PCSs targeting advanced power management.

Therefore, this paper's central idea is to fulfill a twofold need: on the one hand, the design of a capable PCS architecture optimized for handling a per-core, reactive thermal and power control strategy within the required power budget and timing deadlines. On the other, the design of an agile, closed-loop framework for the joint HW/SW development of integrated control systems, leveraging the capabilities of modern HeSoC platforms.

To the best of the authors' knowledge, this work proposes the first research platform where the full-stack (HW, real-time OS (RTOS), PCF and power/thermal emulation setup) required for low-level controller (LLC)-driven on-chip power and thermal management is released open-source.[3] Unlike traditional Linux-capable symmetric multiprocessing (SMP) multi-core systems, as a research platform for on-chip power and thermal management, ControlPULP aims to keep the design's HW and SW complexity to a 32-bit manager and 32-bit programmable many-core accelerator (PMCA) with RTOS support. This design choice better fits the resource constraints of a controller embedded in the *uncore* domain of a larger CPU that is assumed to integrate out-of-order, massive application-class processing elements already.

The manuscript significantly extends the conference work in [17] by detailing and jointly validating the power/thermal model on the HeSoC-FPGA with the PCS and PCF. Overall, the work provides the following contributions:

1. We design an end-to-end RISC-V parallel PCS architecture named ControlPULP, based on open RISC-V cores and hardware IPs [18]. ControlPULP is the first fully open-source PCS with a configurable number of cores and hardware resources to track the computational requirements of the increasingly complex power management policies of current and future high-performance processors (Sect. 3), as well as the first proposed in the RISC-V community. ControlPULP integrates a *manager core* coupled with a multi-core PMCA (cluster) with per-core FPUs for reactive control policy step computation. The cluster subsystem hosts a direct memory access (DMA) engine to accelerate the data transfers from on-chip sensors. This enables data acquisition with 2D stride patterns, a crucial capability when reading from Process, Voltage, Temperature (PVT) sensors with equally spaced address mapping (Sect. 5.3).
2. We tailor ControlPULP to meet real-time power management requirements. The architecture integrates a fast Core Local Interrupt Controller (RISC-V CLIC [19]) tasked to process the interrupt messages associated with OS- and BMC-driven commands, as well as a low latency predictable interconnect infrastructure (Sect. 5.3.2).
3. We demonstrate the end-to-end capabilities of ControlPULP with a case study on the control algorithm quality of service (QoS). We develop a closed-loop evaluation framework based on the concept of heterogeneous system on chips

---

[3] https://github.com/pulp-platform/control-pulp.

Table 1 Comparison among existing proprietary and freely-available PCS from industry and academia

PCS	Provider	HW	PCF scheduling	Programmable accelerator	Openness (HW/SW)
Industry					
PCU	Intel	1-core, HW FSMs	n.a.	✗	✗✗
SCP, MCP	Arm	2-cores	SW FSMs	✗	✗✓
SMU	AMD	1-core	n.a.	✗	✗✗
OCC	IBM	1-core, microcode engines	SW FSMs	✗	✗✓
Academia					
Bambini et al. [20]	Academic	1-core	RTOS	✗	✗✓
This work	Academic	1-core, cluster accelerator	RTOS	✓	✓✓

(HeSoCs) for real-time characterization of the control policy while simultaneously fast-prototyping the underlying hardware (Sect. 4). The framework relies on a *power, thermal and performance model* of the controlled CPU as *plant*, which is paired with workload instruction traces and the PCF to realize the closed-loop. This approach enables a multi-step validation and characterization environment ranging from software in the loop (SIL) and model in the loop (MIL) to hardware in the loop (HIL) abstractions (Sect. 5.5).

4. Finally, we benchmark the reactive control algorithm against one of the few freely-accessible industry-graded state-of-the-art (SOTA) control policy, IBM's OpenPower, in a MIL simulation, achieving 5% more precise set-point tracking (Sect. 5.5).

2 Related Work

There is little publicly available information on PCSs architectures and their HW/SW interface. Table 1 summarizes the leading solutions from academia and industry.

Intel's Power Control Unit (PCU), introduced with Nehalem microprocessor [21], is a combination of dedicated hardware state machines and an integrated microcontroller [8]. It provides power management functionalities such as dynamic voltage and frequency scaling (DVFS) through voltage-frequency control states (P-states and C-states), selected by the HW (*Hardware-Managed P-States*). The PCU communicates with the PEs with a specialized link through *power management agents*. Intel's main control loop runs at 500 μs [22].

AMD adopts a multiple power controller design, with one System Management Unit (SMU) for each CPU tile (group of cores) in a Zeppelin module. All System Management Units (SMUs) act as slave components, monitoring local conditions and capturing data. One of the SMUs also acts as a master, gathering all information from the slave components and then choosing the operating point for each core [23].

Arm implements two independent PCSs based on the Arm Cortex-M7 micro-controller, System Control Processor (SCP) and Manageability Control Processor (MCP). The SCP provides power management functionality, while the MCP supports communications functionality. In Arm-based SoCs the interaction with the OS is handled by the System Control and Management Interface (SCMI) protocol [24]. SCMI provides a set of OS-agnostic standard SW and HW interfaces for power domain, voltage, clock, and sensor management through a shared, interrupt-driven mailbox system with the PCS.

The IBM On-Chip Controller (OCC), introduced with Power8 microprocessor, is composed of 5 units: a central PowerPC 405 processor with 768 KiB of dedicated SRAM and four microcode general-purpose engines (GPEs). The latter are responsible for data collection from PVT sensors, performance state control and CPU stop functions control (*PGPE* and *SGPE*) respectively. IBM OCC's PCF is called Open-POWER, and has a periodicity of 250 µs [15]. It relies on a *frequency voting box* mechanism to select a frequency for each core conservatively based on the minimum input—highest *Pstate*—from several independent power-control (*control vote*) and temperature-capping (*thermal control vote*) features. The *thermal control vote* consists of one proportional integral derivative (PID) that reduces the frequency of each core based on the temperature of the hottest processing element (PE). Furthermore, similarly to Arm's SCMI standard, IBM's OCC relies on a *command write attention/interrupt mechanism* to notify the *PGPE* of an incoming asynchronous command/request to be processed[2], such as the desired *PState*. *PGPE* arbitrates this information with the voting box output from the PowerPC 405 according to a minimum *PState* policy.

Last, Bambini et al. [20] show that a single-core, RISC-V-based microcontroller can execute similar reactive control algorithms with a control loop of 500 µs. The work relies on the SPI interface to conduct the main periodic task and lacks support for essential HW intellectual propertys (IPs) such as DMA and floating point unit (FPU).

All the SOTA power controllers lack the flexibility and scalability of a multi-core architecture supported by adequate I/O bandwidth from/to on-die and off-die power monitors and actuators coupled with fast interrupt handling hardware for HLCs (OS and BMC) commands dispatching, which is the critical innovation provided by ControlPULP.

It is essential to notice that PCSs design is only half of the coin, as its performance depends on the PCF and real-time performance achieved in closed-loop. Several works have targeted the emulation of 'in-field' power management algorithms, but none of them has validated the PCS and PCF designs jointly (co-designed).

Atienza et al. [25, 26] propose a thermal emulation framework where a generic MPSoC is implemented on FPGA. A host computer executes the *thermal model*, which is driven by real-time statistics from processing cores, memories, and interconnection systems emulated on the FPGA. With the increasing number of cores in modern CPUs, an FPGA approach that implements the entire MPSoC is not feasible and incurs resource partition with high platform costs. Beneventi et al. [10] design a similar closed-loop approach where a subset of the Intel Single-Chip-Cloud computer's PEs execute the *thermal model* while receiving

online workload information from the remaining PEs. While employing actual and commercial hardware in the emulation, the work focuses on the high-level controller (HLC) only and with a software-centric methodology, being HW/SW co-design of the LLC prevented by the inaccessibility of the underlying HW.

Beyond these approaches, our work aims at co-designing the LLC (PCS) and HW/SW components of the controlled plant to assess "in-field" performance. This step is well understood in the design flow of classic control systems targeting automotive, avionics, and robotics domains, where progressively more realistic simulations of the plant in the closed-loop are coupled with the introduction of the actual hardware controller (HIL) and checked against model-based closed-loops (SIL, MIL) from early development stages of the control design.

A taxonomy of the various design possibilities for a control system that adapts well to the scenario of this work is provided in [27]. On the controller side, an FPGA emulation approach provides the benefit of testing the control firmware developed during SIL/MIL on the actual hardware controller with the guarantee of cycle-accurate simulation. The latter is required to achieve fine-grained hardware observability and controllability [16] and one-to-one correspondence between the register transfer level (RTL) source and its FPGA mapping in terms of clock cycles (Direct FPGA Architecture Model Execution (FAME) systems [28]). On the plant side, a virtual simulation of the plant is preferred to an FPGA-based approach, which incurs high development costs due to the complexity of the whole plant system to be emulated. For this purpose, MCUs are the solution adopted by the industry nowadays: they are cheap, general-purpose and with a standardized and automated software development process.

The combination of MCUs and field-programmable gate arrays (FPGAs) flexibility takes the best of both worlds and is supported by modern, commercial off-the-shelfs (COTSs) FPGA-SoCs platforms such as Xilinx Zynq Ultrascale+ and Versal families, making it the solution of choice adopted in our work.

3 The ControlPULP Platform

ControlPULP extends commercial controllers' single-core architecture, providing the first multi-core RISC-V PCS architecture. To make the paper self-contained, in the following, we first provide a high-level overview of a generic high performance computing (HPC) CPU that integrates the PCS. We detail the platform's software stack, which helps understand the control policy flow, the interaction between the controlled plant and the controller, and the terminology adopted throughout the manuscript (Sect. 3.1). Finally, we detail ControlPULP's hardware architecture and design trade-offs (Sect. 3.2) to implement the control algorithm.
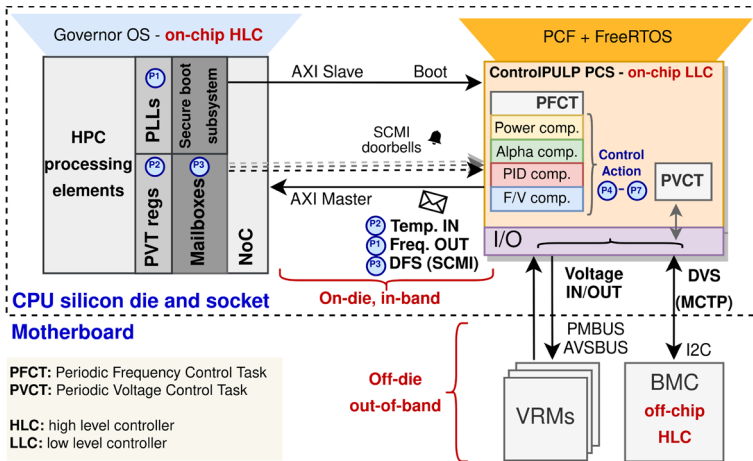
Fig. 1 High-level overview of the system. We highlight on-chip and off-chip HLCs (OS and BMC), the LLC (ControlPULP) and the MIMO IO interfaces. Furthermore, the figure details the PCF phases described in Sect. 3.1

Fig. 2 ControlPULP software stack. The application control policy (PCF) executes on top of FreeRTOS, which controls the hardware with target-specific drivers and hardware abstraction layer (HAL) application programming interfaces (APIs)
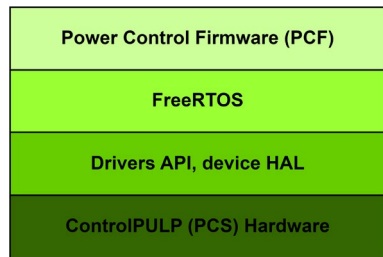
3.1 Controlled Plant and Power Control Firmware

Figure 1 depicts the high-level structure of the HPC CPU silicon die. From a control perspective, we distinguish between the controlled system (*plant*) and the on-chip LLC controller, namely ControlPULP. Furthermore, the figure illustrates the environment surrounding the CPU socket hosting the silicon die, namely the motherboard, with off-chip actors, such as VRM and BMC. The OS running on the PEs, as well as the off-chip BMC, are the two HLCs.

We assume the controlled plant is a many-core HPC CPU with 72 application-class PEs, and exposes hardware mailboxes through a network on chips (NoC) interconnect system. While the mailboxes mediate the dynamic frequency scaling (DFS) commands (e.g. target frequency) dispatched on-chip by the governor OS HLC to the LLC, power management protocols such as Power Management Bus (PMBUS), Adaptive Voltage Scaling (AVSBUS) and Management Component Transport Protocol (MCTP) mediate dynamic voltage scaling (DVS) commands (e.g. power budget threshold) from the BMC HLC, as detailed in Sect. 3.2.3.

To clarify the terminology in Fig. 1, *on-die* designates any element of the HPC CPU that resides on the chip die, such as PVT sensors and registers, frequency actuators, and mailboxes. *In-band* services refer to SCMI-based interaction and PVT data acquisition. Lastly, *off-die* indicates VRM communication and BMC requests through *out-band* services.

The PCF executes the thermal and power control functions and manages on-die and off-die communications and data transfers. Real-time priority-driven scheduling with static task priorities and preemption is required to manage the control functions. In this work, we use FreeRTOS, an industry-grade, lightweight, and open-source operating system for microcontrollers. The software stack of the proposed platform is shown in Fig. 2.

The PCF routine implements a two-layer control strategy [20], managed by the FreeRTOS scheduler as two periodic tasks characterized by multiple harmonic frequencies: the **periodic frequency control task (PFCT)**—2 kHz, i.e. 500 µs—and the **periodic voltage control task (PVCT)**—8 kHz, i.e. 125 µs. Splitting the control routine into two tasks grants more fine-grained control actions and helps meet different performance requirements and sensors-update frequencies. For instance, power changes rapidly due to instruction-level variation of the effective switching capacitance of the computing units, while temperature variations are slower. The control policy has to handle these widely split time scales. Furthermore, VRM generally update more frequently than temperature sensors.

The PFCT is the main control task. It receives the desired clock frequency operating point for each processing element from the OS HLC governor as well as a power budget threshold from the BMC HLC via the PVCT, and computes the optimal frequency/voltage pair to meet the physical and imposed constraints of the system. The task is then responsible for dispatching the controlled frequency by directly accessing the CPU PLLs, as from Fig. 1. The PFCT executes a two-layer control strategy [20] consisting of a *power dispatching layer* and a *thermal regulator layer*. PFCT's control step *n* comprises several phases, illustrated in Fig. 1: **(P1)** allocate the controlled clock frequency computed at step $n-1$ to each core; **(P2)** read the PVT sensor's registers and the workload characteristics from each core; **(P3)** obtain commands and information on the constraints (DVFS operating points, power budget) from the OS and the BMC; **(P4)** compute the estimated power for each core and the total consumed power of the system; **(P5)** apply a power capping algorithm, such as *alpha* [20] when the total power exceeds the power budget constraint; **(P6)** further reduce the power of each core through proportional integral derivatives (PIDs) computation when the temperature at phase (P2) exceeds the threshold; **(P7)** compute both the frequency and voltage to be applied to the controlled system. Throughout this manuscript, we name *control action* the computational body of the PCF execution (P4)–(P7).

The PCF does not provide per-core voltage control but groups PEs in coarse-grained voltage domains. The PVCT is responsible for detecting the changes in the system's power consumption. It periodically reads the power consumption of the voltage rails—associated with each voltage domain—from the VRM and programs micro-architectural power/instruction throughput capping interfaces (if supported by the PEs). Lastly, it modifies the PFCT's power budget threshold as requested by the
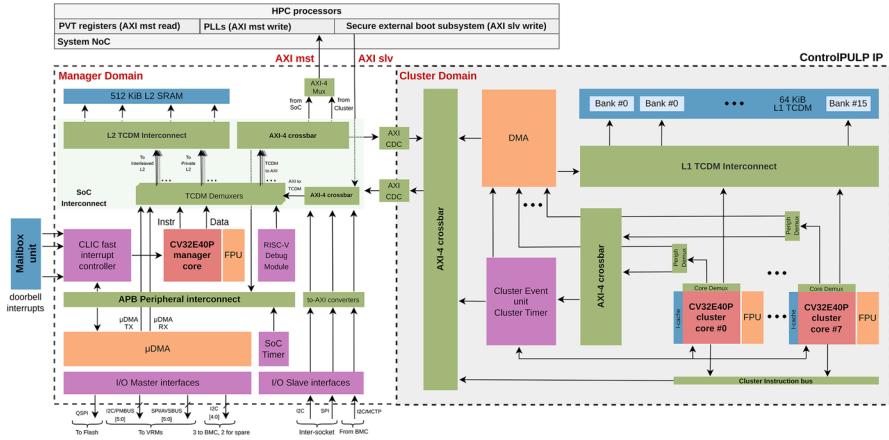
Fig. 3 ControlPULP hardware architecture. On the left, the *manager domain* with the *manager core* and surrounding peripherals. On the right, the *cluster domain* accelerator with the eight cores (workers)

BMC. Even though the PFCT computes both optimal frequency and voltage, it only dispatches the frequencies to apply at phase (P1) in step $n + 1$. In contrast, the PVCT dispatches the voltages at step n (one iteration before), hence the names chosen for the two tasks.

3.2 System Architecture

Figure 3 provides a block diagram overview of ControlPULP. The top-level subsystem of the design is the *manager domain*, consisting of a CV32E40P open-source[4] industry-grade processor core and a set of system I/O interfaces to interact with external peripherals and memory-mapped devices (Sect. 3.2.3). The primary microcontroller-like subsystem is also a recurrent element in the SOTA designs surveyed in Sect. 2.

3.2.1 Real-Time and Predictability

In the following, we discuss the architectural design decisions concerning RAM banking and interrupt processing taken to make the design more suitable for realtime workloads.

L2 Memory Banks Constant Access Time

The L2 RAM, which is the RAM block connected to the *manager domain* and system I/O interfaces, is sized to 512 KiB, enough to fit the whole firmware binary and data so that no swapping is required. The L2 RAM comprises six banks. The access time to each bank is constant when there are no access conflicts. Two of these banks are marked private to prevent DMA transfers from peripherals and other

components from disturbing the manager core's instruction and data fetching. The manager core has exclusive access to those.

Low Constant Latency Core-Local (CLIC) Interrupt Controller

Provided the need for a real-time and predictable architecture, we extend the original core-local interruptor (CLINT), compliant with RISC-V privileged specifications [29], with the newly proposed core-local interrupt controller (CLIC), currently under ratification by the RISC-V community. We employ an open-source implementation of the CLIC[5] that reflects the latest status of the RISC-V CLIC draft specifications [19]. The integration process includes the addition of CSRs registers in the processor's micro-architecture as per specifications.

The CLIC introduces several improvements to the standard CLINT to achieve faster interrupt handling. Among those are dedicated memory-mapped registers for software configurable interrupt priority and levels at the granularity of each interrupt line, runtime-configurable interrupt mode and trigger type, and support for interrupt preemption within the same privilege level (interrupt nesting). Selective hardware vectoring (SHV) enables the programmer to optimize each incoming interrupt for either faster response (*vectored* mode, when each interrupt service routine has a reserved entry in the interrupt table) or smaller code size (*direct* mode, when each interrupt traps to the same exception handler address). Lastly, with the CLIC, local interrupts can be extended to 4096 lines instead of being limited to the processor's XLEN (32-bit for CV32E40P).

In this work, we implement 256 local interrupt lines coming to ControlPULP from the mailbox infrastructure (Sect. 3.2.3). The CLIC configuration helps reduce the interrupt response latency and is capable of entering the interrupt handler within 30 clock cycles (Sect. 5). This is a crucial property to increase responsiveness on external, agent-driven requests.

3.2.2 Cluster Accelerator

To meet the computational demands of the control algorithms, in particular, when scaling to a large number of controlled high-performance PEs and improving the control performance, we opt for a flexible programmable accelerator, namely a cluster of RISC-V CV32E40P cores—referred to as *workers* in this manuscript—tightly coupled to 64 KiB RAM (L1) and a DMA engine. The accelerator is represented in Fig. 3 as *cluster domain*.

Multi-core Computing System

Control algorithms (Sect. 3.1) can be parallelized on the cluster domain (Sect. 5.2). This guarantees a high grade of flexibility on the software development side, and is in sharp contrast with hardwired control logic featured in SOTA controllers (Sect. 2), which lack flexibility. The *manager core* offloads the control algorithm to the team of workers in the cluster. Each worker features a private instruction cache for instructions stored in the main L2 memory and accesses L1 through a single-cycle latency logarithmic interconnect.

---

[5] https://github.com/pulp-platform/clic.

In the most straightforward parallelization scheme, a worker computes the control action (Sect. 3.1) for a subset of the controlled cores. The number of workers in the cluster is parametric. In the following, we consider eight cores to demonstrate scalability. Each core in the cluster features an FPU with a configurable number of pipeline stages. In our instantiation, we use one internal pipeline stage, which is sufficient to meet our frequency target (Sect. 5.1). Furthermore, Montagna et al. [30] show that this configuration achieves high performance and reasonable area/energy efficiency on many benchmarks.

2-D DMA Transfer Engine

The *cluster domain* integrates a multi-channel DMA with direct access to L1 RAM and low-programming latency (62 clock cycles, Sect. 5.3). The DMA's main task is to provide direct communication between L2 and L1 memories in parallel and without intervention from the *manager* or *cluster* domains [31].

We tailor the DMA's capabilities to suit the control policy use case by (i) directly routing the cluster DMA to the PVT sensors registers through the outgoing AXI master interface, which guarantees flexibility by decoupling data transfers and computation phases, (ii) exploiting 2-D transfers for equally spaced PVT registers accesses and (iii) increasing the number of outstanding transactions (up to 128) to hide the latency of regular transfers.

Commercial PCSs described in Sect. 2 also decouple the actual computation from data acquisition. For instance, according to available SOTA information, IBM's OCC employs general-purpose cores (GPEs) tasked to read PEs's data and temperatures instead of a dedicated data mover engine with reduced programming interface overhead [15].

3.2.3 System I/O Interfaces

AXI4 Interfaces ControlPULP features two AXI4 ports, one *master* and one *slave*, with 64-bit W/R, 32-bit AW/AR wide channels. They play a crucial role in the design and guarantee low-latency communication with the controlled system. The AXI slave drives the booting process of the PCS. In the high-level structure depicted in Fig. 1, an external, secure subsystem is responsible for loading the PCF binary into ControlPULP's L2 SRAM by mapping to a region of the L2 SRAM. The AXI master is the transport layer over which the PCS collects PVT sensors data and power policy target requirements. On the other hand, it dispatches the computed optimal operating point to the PEs during the control policy (Sect. 3.1). Its address space is reachable from both the *manager* and *cluster* domains.

Mailbox-Based SCMI Communication

ControlPULP adopts and implements the Arm standard SCMI protocol to handle external power, performance, and system management requests from the OS HLC. SCMI allows an OS kernel that supports SCMI to interact with Control-PULP without requiring a bespoke driver directly. Furthermore, the design of the SCMI protocol reflects the industry trend of delegating power and performance to a dedicated subsystem [11]. SCMI involves an interface channel for secure and non-secure communication between a caller (named *agent*, i.e., one processing element of an HPC CPU) and a callee (named *platform*, i.e., ControlPULP).

The latter interprets the messages delivered by the former in a shared memory area (mailbox region, Fig. 1) and responds according to a specific protocol. The proposed PCS implements an interrupt-driven transport mechanism through the CLIC. In our use case with 72 controlled cores, the platform can process up to 144 secure, and non-secure interrupt notifications.

We design hardware mailboxes as the transport layers for the SCMI communication mechanism. The shared memory region is implemented according to the single-channel memory layout described in the specifications. We reserve a space of 8B for the implementation-dependent payload field, totaling 40B per channel. Each channel is associated with one outgoing interrupt line in the agent-platform direction (doorbell). The agent identifier is encoded with the message such that more agents can use the same channel as the notification vehicle. Hence, the number of interrupts dispatched by the mailbox system can be smaller than the number of agents, with the benefit of reducing the area of the interrupt controller's configuration register file. The agent identifier is not described in the official specifications and is mapped to a reserved field of the single-channel memory layout. With 64 channels as employed in this work, the shared memory region has a size of about 2kiB.

I/O Peripherals for Voltage Rails Power Management ControlPULP integrates a peripheral subsystem in the *manager domain*, where an I/O data engine unit (named μDMA intellectual property (IP)) enables autonomous communication between off-die elements and the L2 SRAM with low programming overhead for the manager core. In this work, we upgrade the peripheral subsystem with industry-standard power management interfaces to handle off-die communication services. The PCS integrates 6 AVSBUS and PMBUS interfaces towards VRM. The PMBUS and AVSBUS bus protocols extend I2C and SPI to monitor voltage and power rails digitally, preserving optimal speed/power consumption trade-off. 5 I2C master/slave interfaces manage the communication with the BMC and other board-level controllers. The slave interfaces transfer DVS operating points (power budget) from the BMC according to the Platform Level Data Model (PLDM) and MCTP transport layer protocols.

4 FPGA-Based HIL Thermal, Power, Performance and Monitoring Emulation Framework

This section provides an end-to-end description of the HeSoC based HIL emulation methodology. The closed-loop approach that we implement introduces the actual LLC executing the PCF as ISUT and relies on a *thermal, power and performance model* for the controlled CPU plant. While the control literature refers to such a setup as processor in the loop (PIL) or FPGA in the loop (FIL), others already define a closed-loop as HIL when the connection between the integrated system under test and the plant reflects the actual hardware interface of the final manufactured silicon, without relying on a virtual representation [16]. Since this is the case for the present work, we adopt the HIL terminology.
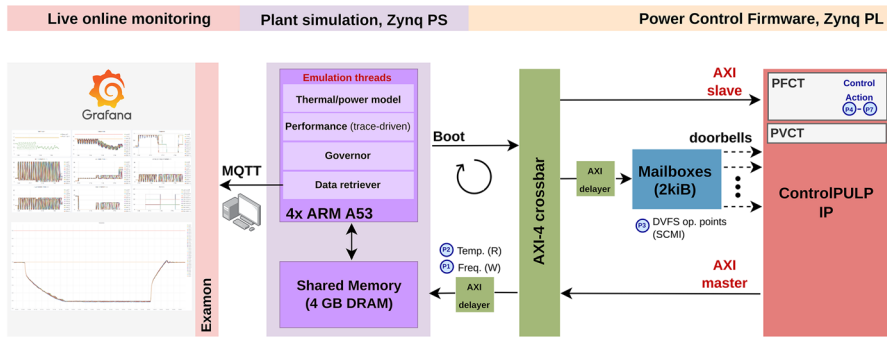
Fig. 4 HIL test procedure on an FPGA-SoC with ControlPULP

Table 2 ControlPULP resource utilization on the Xilinx UltraScale+ ZCU102

Resource	Utilization	Available	Utilization [%]
LUT	265718	274080	**96.95**
FF	153155	548160	27.94
BRAM	278	912	30.48
DSP	93	2520	3.69
I/O	83	328	25.30

Bold value indicates the number of LUTs

4.1 HIL System and PCS Mapping on FPGA

The Xilinx Ultrascale+ FPGA family is widely adopted in the heterogeneous computing domain [32]. It features a Processing System (PS), or host computer, and a Programmable Logic (PL), namely the configurable FPGA, integrated on the same physical die. Figure 4 provides an overview of the FPGA based emulation framework designed in this work and its main actors.

The PS consists of an industry-standard, application class, quad-core, 64-bit Armv8® Cortex-A53 *Application Processing Unit* with 32 KiB L1 instruction and data cache per core and a 1 MiB L2 cache shared by all four cores—on-chip memory (OCM)—, clocked at 1.2 GHz, a dual-core Cortex-R5F *Real Time Unit*, and Mali™-400 MP2 GPU based on Xilinx's 16nm FinFET.

We implement ControlPULP on the ZCU102 PL with Xilinx Vivado 2022.1. The PS interfaces with ControlPULP through the AXI4 master and slave ports (Sect. 3) and provides it with an external reset and 125 MHz clock, which is internally converted to 50 MHz system clock by the ISUT. Table 2 shows the board's resource utilization inferred by the implementation. Overall, ControlPULP fits the available resources with almost 97% utilization.

The communication between the plant simulation and the ISUT is regulated through a 4 GiB off-chip DDR4 DRAM provided on the board, which allows

them to exchange data and realize the feedback loop with up to 19.2 GB/s of bandwidth.

4.2 Plant Simulation

In the considered control scenario, the plant provides a thermal, power, performance, and monitoring framework capable of simulating the power consumption and temperature of a high-end CPU processor. The plant simulation is programmed in *embedded C*. Its functionalities are split into several threads employing `pthread` Linux libraries to exploit multi-core parallelism. In the following, we refer to Fig. 4 and detail the threads' organization and interaction with the ISUT.

The ***thermal and power model thread*** is the leading simulation thread. It carries out the computation of the thermal and power model with a periodicity of 1 µs which, according to the literature [33, 34], is fast enough to capture the thermal dynamics, i.e., three orders of magnitude faster than the fastest thermal time constant $\sim 1ms$, and to simulate power spikes and oscillations that are not filtered by the hardware power delivery network (up to $\sim 100\,\mu s$ [35]). The thread takes as input the values controlled and dispatched by the PCS controller (frequency and voltage) and the modeled workload from the *performance model thread* to compute the average consumed power and the temperature of each core. The average consumed power is computed with an algebraic model of the cores, which includes core-to-core variability and noise, according to [34]:

$$P_i = P_{i,\text{static}} + P_{i,\text{dynamic}} = \kappa(T_{\text{Si},i}) \cdot (I_{\text{cc},i} \cdot V_{\text{dd}} + (C_{\text{eff},i} \cdot f_i \cdot V_{\text{dd}}^2)) \tag{1}$$

where κ represents the temperature dependency of the computed power, and C_{eff} is the equivalent effective capacitance of the controlled CPU. The temperature is simulated through a discrete state space model, which considers the temperature and instantaneous power of the neighboring simulated cores. Coefficients are extracted from a commercial multi-node RISC-V cluster capable of providing an HPC production stack, Monte Cimone [36].

The ***performance model thread*** assists the *thermal and power model thread* by providing the workload characteristics for the next interval of time. We rely on a *trace-driven* approach for the simulated workload for practical reasons: (i) the traces could be extracted from accurate benchmarks such as PARSEC [37], SPEC [38] or real HPC applications; (ii) simpler but effective performance models can be built on top of workload traces, e.g., instructions per cycle (IPC) model and roofline model [39], enabling the evaluation of the impact of the power management policies; (iii) the power consumption is mainly affected by workload composition, i.e., memory bandwidth and vector/single instruction, multiple data (SIMD) arithmetic density.

In this paper, we craft a synthetic benchmark (`Wsynth`) that stresses the control corner cases and consists of maximum power (`WsynthMax`) and idle power (`WsynthIdle`) instructions, mixed power instructions (`WsynthMix`) and lastly instructions with different power densities and fast switching (`WsynthFast`) to stress the power limiter and the shorter timing constants of the temperature response.

The **governor thread** emulates command dispatching agents such as the operating system running on the CPU or the off-chip BMC in the motherboard. They are tasked to send requirement directives to the controller, such as DFS and DVS operating points. In the plant simulation presented in this manuscript, the OS and BMC communication is modeled through the SCMI transport layer. Future work will integrate PLDM/MCTP transport layers supported by ControlPULP (Sect. 3.2.3) in the HIL framework for BMC-related communication.

Finally, the **data retriever thread** collects all the simulation data and periodically dispatches them through the network. It relies on Eclipse Mosquitto as a message broker, which implements the Message Queuing Telemetry Transport (MQTT) network-based messaging protocol. Collected data are fed to *Examon* [40], an open-source[6] framework that enables performance and energy monitoring of HPC systems. *Examon* relies on *Grafana* as interactive data-visualization platform with live-updated dashboards.

4.3 HIL Testing Procedure

We identify two main phases to carry out the emulation. In the *system setup* phase, the communication between PS and PL happens in a conductor-follower fashion. The PS drives the PCS deployment on the FPGA, its booting process, and firmware binary flashing. The AMBA AXI4 protocol regulates the data transmission across this communication channel. To this aim, we generate a complete embedded SMP Linux system paired with a persistent root file system on top of the four Arm A53 processors with Buildroot.[7]

In the subsequent *system emulation* phase, PS and PL execute the respective routines and communicate *asynchronously* with each other through the shared DRAM memory region. The lack of an explicit synchronization point between the controller and the simulated plant is inherent to the nature of the control since dynamic thermal and power management involves run-time active control. At the same time, the underlying MPSoC varies its workload to meet a specific computational need.

The *system's setup and emulation* phases consist of the following steps:

1. Linux First Stage BootLoader (FSBL): the PL is programmed with ControlPULP's *bitfile*, which contains the hardware design information of the controller.
2. Linux *U-Boot*: Linux kernel boots on the Arm Application Processing Unit (APU) cores.
3. ControlPULP is clocked from the PS, out of reset, and in idle state. Internal divisions of the external clock are handled within ControlPULP.
4. The PS drives ControlPULP booting process by flashing the L2 SRAM with the control firmware executable (Fig. 3).
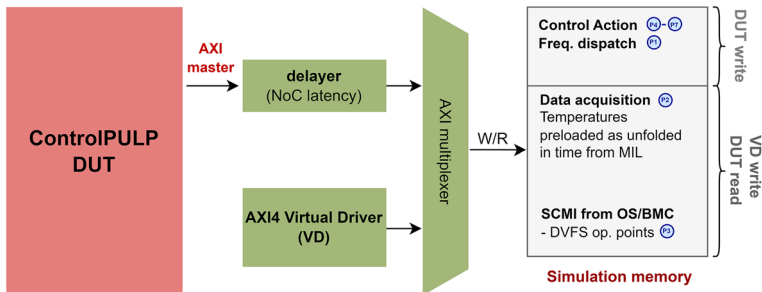

Fig. 5 ControlPULP RTL testbench simulation environment

5. The PS and PL start to asynchronously execute the plant simulation and the control firmware routines, respectively, through the shared memory.

The data are then collected for online dashboard monitoring, as detailed in the previous section.

The key benefit of the proposed methodology is the flexibility gained at the HW/SW interface. Design space exploration can be carried out on both the hardware controller and the control algorithm, with a short turnaround development time. The methodology especially fits integrated (on-chip) control systems validation and co-design due to the native on-chip hardware flexibility offered by the SoC-FPGA ecosystem.

5 Evaluation

In this section, we analyze and characterize both hardware (the ControlPULP platform) and software (the PCF) layers:

- We break down ControlPULP's post-synthesis area, which represents a small overhead ($< 1\%$) compared to a modern HPC processor die (Sect. 5.1).
- We first evaluate ControlPULP architecture with a cycle-accurate RTL testbench environment as depicted in Fig. 5. We model the latency of the interconnect network sketched in Fig. 1 by adding a programmable latency to the AXI4 interface. In the described test scenario, we first study the parallelization of the *control action* (P4)–(P7) on the cluster (Sect. 5.2). We then characterize *in-band* transfers, namely strided DMA accesses for data acquisition from PVT registers and CLIC interrupt latency with SCMI command processing (Sect. 5.3). Finally, we show the overall performance improvement of a single control step when accelerating control tasks in the cluster compared to single-core (Sect. 5.4). The testbench depicted in Fig. 6a does not provide the RTL description of the surrounding HPC processor. Instead, we model the closed-loop with a shared memory region between the PCS platform—the ISUT—and the system under control. Note that the real-time temperature and telemetry information from the

Table 3 ControlPULP post-synthesis area breakdown on GF22FDX technology

Unit	Area [mm^2]	Area [kGE]	Percentage [%]
Cluster domain	0.467	2336.7	25.5
Manager domain	0.135	675.9	7.4
L1 SRAM	0.119	595.7	6.5
L2 SRAM	1.108	5542.1	60.6
Total	1.830	9150.3	100

HPC processor are pre-computed from a MIL emulation of the control algorithm executed with a fixed time step and statically stored in the simulation memory as unfolded in time.

- Since the standalone RTL simulation environment fails to provide a near-real-time closed-loop emulation framework for the control algorithm, we rely on the FPGA-SoC methodology detailed in Sect. 4 to analyze the PCF QoS when ControlPULP is mapped on real hardware and compare against a pure MIL approach to assess the validity of the HW/SW co-design.
- Finally, we show that the proposed PCF compares favorably against one of the most well-documented and freely accessible SOTA industrial solutions on the market, IBM's OpenPower (Sect. 5.5). The comparison is carried in pure MIL simulation, being IBM's PCS hardware source publicly unavailable.

5.1 Area Evaluation

We synthesize ControlPULP in GlobalFoundries 22FDX FD-SOI technology using Synopsys Design Compiler 2021.06. One gate equivalent (GE) for this technology equals 0.199 μm^2. As shown in Table 3, the design has an overall area of 9.1 MGE, 32% occupied by the computing cluster accelerator and local L1 memory, and almost half (60%) by the main L2 memory in the *manager domain*. We use a system clock frequency of 500 MHz. The rationale of the frequency choice is the following: the PCS typically belongs to the same clock domain of the CPU's *uncore* region. While the *core* region hosting the application-class PEs runs at frequencies in the order of 1-3GHz, the *uncore* is clocked at lower frequencies, in the order of hundreds of MHz, with modern architectures and technology nodes.

The target controlled-system die area is assumed comparable to other commercials, multi-core (> 64) server-class processors such as [3], about 460 mm^2. By correlating the gate-equivalent count of the HPC CPU die in the same technology node of this work, ControlPULP would still represent about 0.5% of the available die area.[8] This first-order estimation makes the design choice of a parallel PCS valuable since its capabilities are much increased, while the silicon area cost remains negligible within a high-performance processor die.

---

[8] This has to be considered a first approximation, since it compares post-synthesis results with publicly available data of a modern HPC die, nowadays manufactured in a more advanced technology node.

(a)

(b)

(c)

(d)
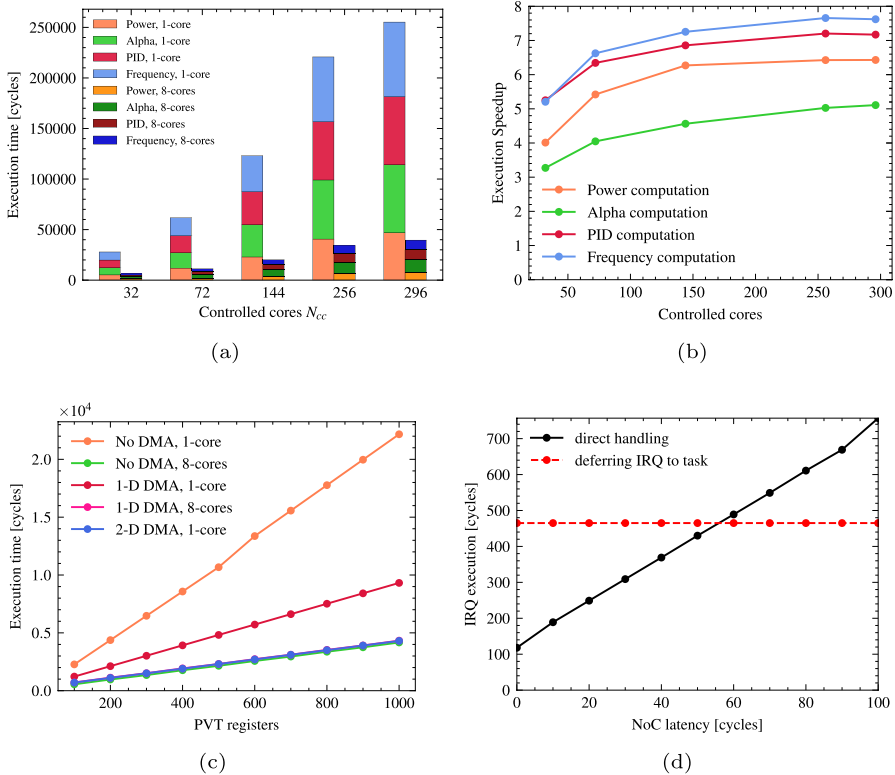
Fig. 6 **a**, **b** Firmware *control action*, execution time, and speedup comparison between single-core (*manager domain*) and 8-cores (*cluster domain*). **c** *in-band* data acquisition from simulated PVT registers, execution time without and with DMA in 1-D and 2-D configurations. **d** Execution time in the interrupt handler from the interrupt edge to its completion with a basic SCMI message at varying interconnect network access latency to the mailbox

5.2 Firmware *Control Action*

In the following, we analyze the execution of the PCF phases (P4)–(P7) on the multi-core cluster accelerator. We enforce power capping (*alpha* reduction [20]) to evaluate each computational phase fairly. Each cluster core is responsible for a subset of the controlled PEs. The parallelization is implemented as a fork-join process where the workload is statically distributed among the workers. In ControlPULP, the construct is implemented through a per-worker `thread_id` $\in [0 : N_{workers} - 1]$ and an equally distributed `chunk_size` where $\texttt{chunk\_size} = \dfrac{N_{ctrl\_cores}}{N_{workers}}$. We are interested in extracting performance figures for the *control action* in a single periodic step n. We execute the PCF for S steps to amortize the effect of the initially cold instruction cache. Finally, we perform the arithmetic mean over S to get the mean absolute execution time for each (P4)–(P7) phase.

We report the execution time τ_0 and the multi-core speedup ($\frac{\tau_{0,single}}{\tau_{0,multi}}$) at varying number of controlled cores N_{cc} for each PCF phase in Fig. 6a and b respectively. The

Table 4 Interrupt latency from interrupt edge to the first instruction in the interrupt handler as number of cycles

Location	Increment [cycles]	Sum [cycles]
CLIC input to output	1	1
CLIC output to core (handshake)	2	3
Claim interrupt	1	4
Jump in vector table to CLIC handler	2	6
Save caller save regs (`addi` + 15 regs)	17	23
Compute and load CLIC handler address	5	28
Jump to CLIC handler address	2	30
Summary	–	30

total speedup of the full *control action* at fixed N_{cc} is the geometric mean over the speedups of each phase. In our use case of 72 controlled PEs, ControlPULP executes the *control action* 5.5x faster than in single-core configuration, reaching 6.5x with 296 controlled cores.

We make the following observations. First, multi-core speedup scales with the number of controlled cores due to the increased workload and is affected by the workload characteristics of each phase. Second, the *control action* is not a fully computational step. In fact, instruction branching associated with power and frequency bounds checks per core introduces additional load/store stalls due to data access contention in a multi-core configuration. Finally, the computational body of (P6) and (P7) can be separated into independent parallel tasks and is thus an embarrassingly parallel problem. Instead, (P4) and (P5) show dependency across the values computed by the workers in the form of reduction sums, i.e., in (P4) to calculate the total power of the CPU and (P5) to calculate a normalization base for *alpha* power capping [20] and again the total CPU power. When a reduction sum is needed, we use a hardware barrier to synchronize the threads and join the concurrent execution on the cluster master core (core 0), which carries out the reduction.

As discussed in the analysis above, the increased parallel compute capability of handling the control's computational workload, paired with the general purpose nature of the accelerator, enable us to (i) improve the control performances with more advanced algorithms and (ii) be fully flexible when designing the control algorithm.

5.3 *In-Band* Services

5.3.1 PVT Sensors

To assess *in-band* services involving PVT physical sensors—phases (P1) and (P2)—, we measure the transfer time required for reading data bursts on the AXI4 master bus with the SoC timer. The exploration is three-fold: (i) direct data gathering from the ControlPULP cluster's cores, (ii) data gathering by offloading the transfers to the DMA in 1-D configuration, and (iii) DMA offload in 2-D configuration [31]. For (i) and (ii), we investigate the data collection on either 1-core or

8-cores configurations. The address range is equally distributed among the issuing cores in the latter scenario. In (iii), one core performs the read operation to highlight the advantages of offloading a single, large transfer with non-contiguous but uniformly spaced addresses to the DMA, which increases the addresses by the selected stride. This configuration becomes important when atomically gathering PVT information from equally spaced address locations (HPC PEs) with only one transfer request. As in Sect. 5.2, we use synchronization barriers to coordinate the eight cores. Figure 6c reports the execution time τ_1 required for data movement when reading from up to 1000 PVT registers (4B each), an estimate bound given the number of PEs and the information needed from them (P, V, T, i.e., $\geq$ 3, lower bound). Figure 6c shows that the best DMA-based transfers assuming 1000 PVT registers (2-D) are 5.3x faster than single-core direct data gathering.

5.3.2 SCMI and Interrupt Latency

An interrupt-driven (doorbell) transport regulates the communication of DVFS operating points in the agent-platform direction. Table 4 gives an overview of the overall CLIC interrupt latency measured as the number of cycles from the triggering edge in the CLIC to the ISR Handler's first instruction. The configuration of interrupt level, priority, and threshold configuration is handled with memory-mapped and CSR accesses to the CLIC register file and the manager core, respectively. This leads to a lower programming latency than software-driven approaches required in standard RISC-V platform-level interrupt controller (PLIC) or CLINT interrupt controllers.

With a working frequency of 500 MHz, the interrupt latency of one SCMI command coming from the OS HLC controller governor is negligible compared to the period of the PFCT that executes every 500 µs, namely 0.01%.

Analogously, the context switch time needed to preempt the PFCT with the PVCT, which runs 4 times faster (every 125 µs), during the execution of a PCF step is 0.08% the available time period of the PFCT, more than enough for the two tasks to coexists while executing their respective policies.

It is essential to notice that the latency of the interconnect NoC between Control-PULP and the mailboxes located in the die (Fig. 1) has a significant impact on the load/store access times, thus the time spent in the ISR, which grows with the interconnect delay size. We show this effect by emulating the shared mailboxes as well as the NoC latency in the RTL testbench environment (Fig. 5). The black line in Fig. 6d reports the execution time for directly decoding and responding to a sample SCMI command (*Base Protocol*, protocol_id = 0x10, message_id = 0x0 [24]) in the ISR when an external simulated driver rings a doorbell to the PCS. The figure reveals that the time spent in the ISR linearly increases with the NoC latency. We tackle the impact of NoC latency by deferring pending interrupts as they are triggered, thus keeping the ISR time short and insensitive to the CPU interconnect network delay, with the FreeRTOS timer API xTimerPendFunctionCall-FromISR(). From the red line in Fig. 6d, we see that deferring interrupt handling

to a task is preferable over direct handling, as it is network-latency insensitive for realistic NoC latencies larger than 50 cycles. Other existing solutions, such as Arm SCP firmware, propose a bespoke *Deferred Response Architecture* [1] to mark selected requests coming from an agent as pending and defer the platform response. We instead rely on a trusted scheduler that decouples OS and PCF driver APIs, improving flexibility and portability.

5.4 System-Level PCF Step Evaluation

We finalize the standalone evaluations of ControlPULP's architectural features from the previous sections with the overall PFCT step cycle count comparison between accelerator-enhanced and single-core configurations, reported in Table 5 in the case of 72 controlled cores. Table 5 shows a breakdown of the required actions.

The total execution time differs in the two execution models. In the single-core case, we execute sequentially with less overhead from data movement

Table 5 Execution time T of a PFCT step, single-core and cluster configurations. SCMI commands exchange and off-die transfers, handled by the SoC *manager core*, are not included in the comparison since they are a shared overhead

Firmware phase	Time step	Execution time [cycles]		Speedup
		1-core	Multi-core	
control action (P4)–(P7)	τ_0	61867	11372	5.5x
in-band transfers (P1), (P2), (P3)	τ_1	5463	3523 (DMA)	1.6x
Offload to the Cluster	τ_2	–	389	–
L2–L1 transfers	τ_3	–	434 (DMA)	–
L1–L2 transfers	τ_4	–	872 (DMA)	–
Return from cluster	τ_5	–	574	–
Step total time	T	67330	13641	**4.9x**

Bold value indicates the total speedup
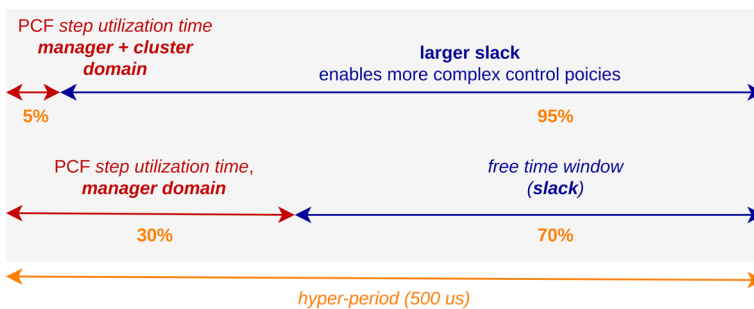
Fig. 7 Benefits of a performant parallel PCS on the control problem, namely, room for more computational-intensive control policies from the concurrent acceleration. The figure shows that the hyper-period free time window (*slack*) increases to almost 95% when the policy is accelerated with the *cluster domain*
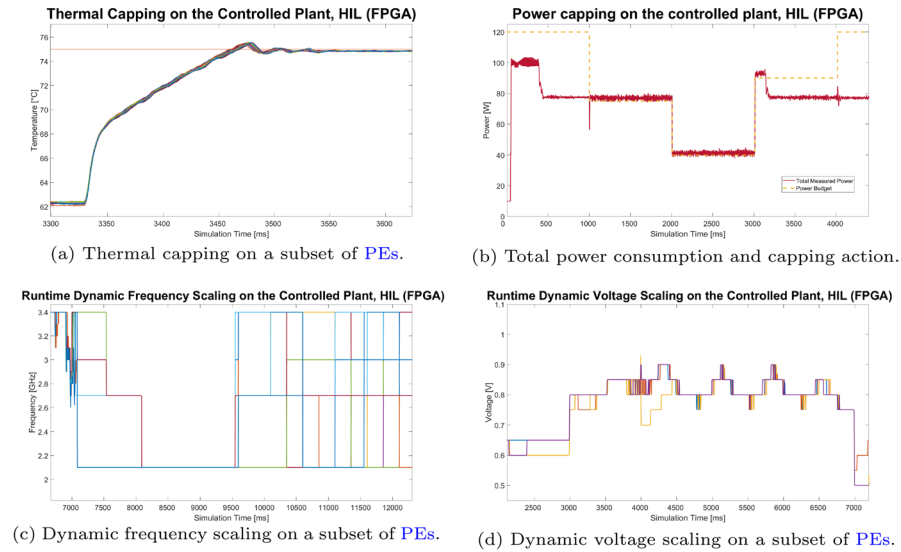
($T_{single} = \tau_0 + \tau_1$). In the multi-core case, ($T_{multi} = max(\tau_0, \tau_1) + \sum_{i=2}^{5} \tau_i$) we (i) execute the computation τ_0 and data acquisition τ_1 at step n concurrently, (ii) rely on $\tau_{0,multi} \ll \tau_{0,single}$, and (iii) introduce an overhead due to additional data movement involving L1 and L2 for data telemetry between *manager* and *cluster domains* during the PFCT.

Overall, multi-core execution achieves a 4.9x speedup over the single-core configuration. Provided a fixed *hyper-period*, i.e., the least common multiple of the control tasks' periods [13], which in this work equals the PFCT step period of 500 µs, Fig. 7 shows the benefits of a programmable accelerator on the control policy time scale. Assuming a working frequency of 500 MHz, and considering the interrupt latency and task preemption context switch time from the scheduler negligible as from Sect. 5.3.2, single-core execution time already leaves a free time window (*slack*) of about 70% the hyper-period with the reactive control algorithm implemented in this work. Cluster-based acceleration significantly raises the free time window to about 95% the PFCT period. This means the acceleration reduces the utilization time to complete the control task within the deadline. Furthermore, the more embarrassingly parallel the control problem, the more the concurrent speedup, hence the benefits of the acceleration.

Other effects in the plant can impact the control loop period in the control scenario presented in this work: (i) phase locked loop (PLL) lock time and voltage regulator response time, (ii) communication latency to sensors/actuators, and (iii) model dynamics evolution. PLL lock time and voltage regulator response time are in the order of microseconds. Similarly, communication latency depends on the interconnect network of the CPU. The PCF computes the frequencies/voltages at time step n and applies them at time step $n+1$. Therefore, latency issues—if any—due to the effects of (i) and (ii) can be addressed by starting early the application of the set points during time step n, provided the control period has enough slack time after the computation phase. Model dynamics (iii) are tightly coupled with the system in use and can be broken down into workload, power, and temperature effects. Temperature variations are in the order of magnitude of milliseconds. Workload and power can potentially change at each clock cycle. Arm proposes the Maximum Power Mitigation Mechanism (MPMM) for mitigating and limiting the execution of high-activity events, such as vector instructions. The management of this mechanism is left to the power controller. This could be a potential extension to ControlPULP as well. The 500 µs hyperperiod used in this work is shorter than the fastest dynamics in terms of temperature, but longer than the workload and power variations. The latter case represents a worst-case scenario to guarantee safe margins on the power constraint in actual HPC benchmarks.

5.5 Control-Level PCF Evaluation

We refer to the PCF QoS as an indicator of the control policy functional correctness (HW and SW) when the simulated MPSoC is assigned a certain workload. In this section, we present the results of this analysis by leveraging the FPGA-SoC

(a) Thermal capping on a subset of PEs.

(b) Total power consumption and capping action.

(c) Dynamic frequency scaling on a subset of PEs.

(d) Dynamic voltage scaling on a subset of PEs.

Fig. 8 Thermal, power capping and DVFS emulation on the FPGA-SoC HIL framework with `Wsynth`. Requests coming from the HLCs (OS or BMC) governors, such as the target frequency and power budget, are processed by the reactive control, phases (P6)–(P7) described in Sect. 3.1. The number of controlled cores is 72

closed-loop framework, the thermal and power model, and the synthetic workload introduced in Sect. 4.3. We subsequently complete the QoS exploration with a comparison against the IBM OpenPower open-source control algorithm.

5.5.1 Standalone QoS Evaluation on the HIL FPGA-SoC Framework

Provided a workload assigned to the simulated MPSoC, the HIL framework enables assessing the evolution in time of critical features of the control algorithm at the granularity of each controlled core (Sect. 3.1): power/thermal capping, workload-aware control, and frequency/voltage set-point tracking as a response to OS and BMC DVFS commands.

Figure 8c and d show the frequency and voltage scaling enforced when the controller's SCMI mailboxes are notified DVFS operating point commands from the plant's *governor thread* running on the PS on a per-PE basis (Sect. 4.2). The SCMI agent dispatches the commands according to the executed workload; interrupts processing is deferred by the FreeRTOS scheduler and committed by ControlPULP's interrupt controller. Once registered, the governor's directive is processed by the PFCT in phases (P3)–(P7) and returns the reduced frequency as computed in the *cluster domain*. Figure 8a and b show per-core thermal capping and the total power consumption and capping in action on the controlled plant during the execution of `Wsynth`, respectively. The maximum thermal limit from Fig. 8a, specific for each PE, is assumed to be 85 °C. The orange line represents instead an additional thermal
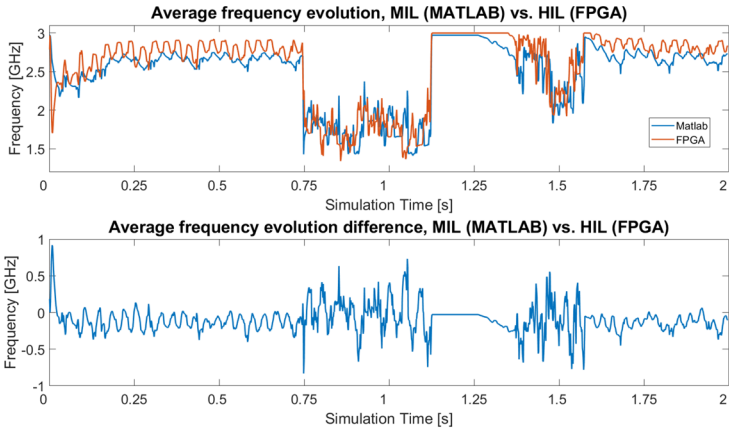
(a) Evolution in time of the average frequency in the simulated tile.
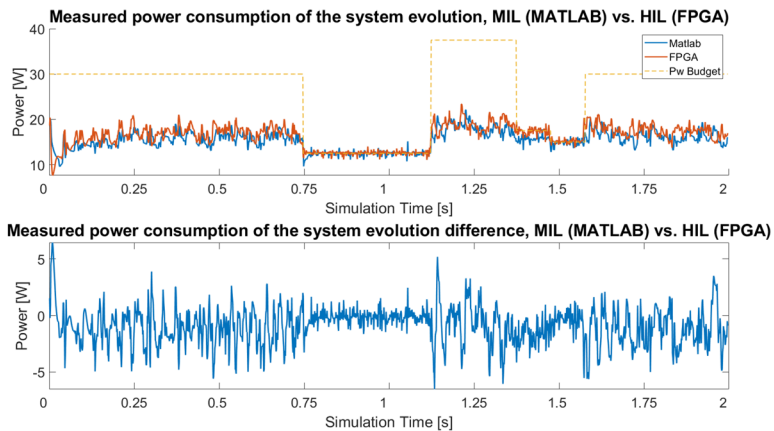
(b) Evolution in time of the average consumed power in the simulated tile. The dashed yellow line represents power budget directives dictated by the HLCs.

Fig. 9 HIL and MIL comparison when `Wsynth` is assigned to the controlled system. The emulation assumes a floorplan with a tile of 9 cores and runs for 2 s to align with MATLAB runtime

threshold required to stabilize the temperature with a safety margin in case of overshoots during the PID. Analogously, in Fig. 8b, the total power consumption of the system is bound by the power budget imposed through SCMI from the HLCs.

The HIL simulation is further compared with the software-equivalent model-based closed-loop from MATLAB Simulink, the first phase of the control algorithm design. Due to MATLAB Simulink runtime execution, we restrict the floorplan of the controlled 72-core CPU to a tile of 9 cores, let the PVCT apply a constant voltage $V_{fixed} = 0.75$ V and fix the simulation duration to 2 s. Figure 9a shows the average frequency evolution in time for the controlled cores in the tile. The discrepancy between the outcome of the two simulations has multiple
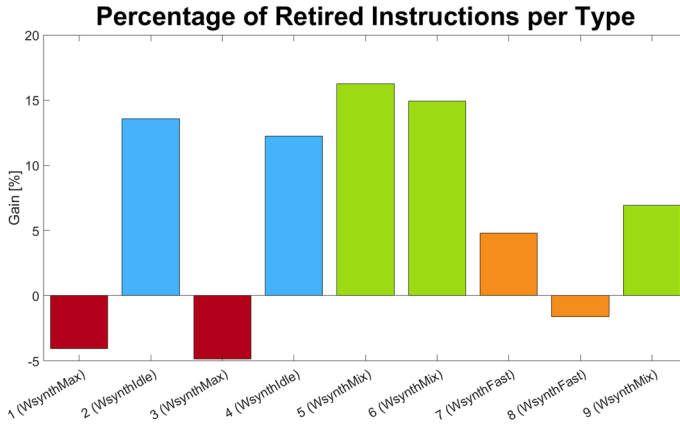
Fig. 10 Comparison between the modified IBM OpenPOWER control with per-core temperature PID for frequency reduction and the original IBM OpenPOWER control. The simulation time is 2 s

reasons: (i) different data collection methodology: MATLAB records data at the exact time, while the FPGA simulation captures the information at a non-deterministic sequence of instants in the simulation interval due to its real-time characteristic, (ii) different resolution, and (iii) uncertainties and non-deterministic control delays introduced in the FPGA emulation and challenging to replicate in the MIL framework. Figure 9b shows the measured consumed power of the assigned workload under power budget constraints from the HLCs, represented with a dashed yellow line. Overall, the HIL-based emulation gives comparable results when validated against the software-equivalent MIL. Albeit power spikes due to the discrepancies described above, DVFS tracking achieves a mean deviation within 3% the system's thermal design power (TDP) (120W in the emulation), more than acceptable for the assessment.

5.5.2 Comparative QoS Evaluation with SOTA

We cross-benchmark the PCF with the IBM OpenPOWER (Sect. 2). We model IBM's control action, i.e., the *Voting Box Control* described in Sect. 2, excluding a few architecture-specific features, and the two-layer PCF control described in this work with MATLAB Simulink to enable a fair and hardware-agnostic comparison. The PID-like coefficients of the IBM control are adapted to the HPC chip model power and thermal characteristics. Wsynth (Sect. 4.2) is distributed as follows: *core 1-core 3* and *core 2-core 4* pairs are assigned WsynthMax and WsynthIdle respectively. *Core 5*, *core 6*, and *core 9* execute WsynthMix while *core 7* and *core 8* are exposed to WsynthFast. We rely again on constant voltage $V_{fixed} = 0.75$ V and do not consider overhead nor delays in the PLLs and VRM operating point transitions. The power budget is changed five times during the simulation to stress all the elements of the control action. The simulation runs for 2 s on a tile of 9 cores.
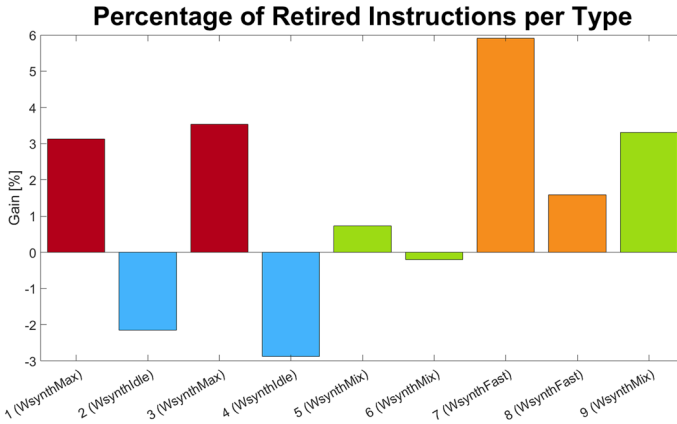
Fig. 11 Comparison between the PCF control and the modified IBM OpenPOWER firmware with per-core temperature PID for frequency reduction. The simulation time is 2 s

First, we show that a controller with a multi-core cluster able to deliver higher computational power is beneficial to the performances of the HPC chip. We compare the IBM control and a version of it with a per-core temperature PID for frequency reduction. In fact, as from Sect. 2, the IBM control policy considers the maximum temperature among the PEs when applying frequency reduction. We conjecture that this limitation is enforced by the limited control policy complexity that can be handled by IBM's OCC. Conversely, ControlPULP enables fine-grained frequency reduction on a per-core temperature granularity. The performance is shown in Fig. 10. The number of retired instructions indicates the execution time achieved by the workload at the end of the simulation: the more retired instructions, the faster the workload, meaning a more efficient control.

While using only one temperature for the whole tile results in an average performance reduction per core of 5%, cores executing high-power instructions (*core 1* and *core 3*) receive a performance increase of 4% and 5% respectively. In fact, being the frequency reduction based on the hotter cores and thus a shared penalty, neighboring cores get colder, and other cores consume less power during power capping phases, leaving more power available to boost performances of *core 1* and *core 3*.

Last, we compare the PCF and the IBM control with per-core temperature PIDs. The PCF control favors cores executing high-power instructions [20] (*core 1* and *core 3* in this simulation with Wsynth benchmark), thus compensating the performance penalty showed in the previous test. Results in Fig. 11 show a performance increase in executed instructions ranging from +2.5 to +5%. This holds for cores with mixed instructions (up to +3.5%) as well, while cores involved in less demanding workloads witness a decrease between −2 and −3%. We conclude that the modified policy with per-core temperature PID calculation can selectively boost the retired instructions, achieving a higher application performance on the HPC chip while still meeting the thermal cap.

6 Conclusion

In this paper, we presented ControlPULP, the first HW/SW RISC-V control system for HPC processors that exploits multi-core capabilities to accelerate control algorithms, featuring a specialized DMA and fast interrupt handling and synchronization. We assess the HW/SW interface and take into account physical state variations surrounding the integrated controller by designing an agile, near-real-time closed-loop emulation framework that takes inspiration from the road paved by modern HeSoC platforms on FPGA. The framework relies on a power and thermal model of the controlled CPU as plant, which is paired with workload instruction traces and the control algorithm to realize the closed-loop. With the proposed multi-core architecture, a control policy step executes 4.9x faster than in a single-core configuration, increasing the hyper-period's *slack* and thus enabling the implementation of more complex and computationally demanding control algorithms (for example, predictive policies) with fine-grained frequency dispatching.

Declarations

Conflict of interest The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

1. Leiserson, C.E., Thompson, N.C., Emer, J.S., Kuszmaul, B.C., Lampson, B.W., Sanchez, D., Schardl, T.B.: There's plenty of room at the top: What will drive computer performance after Moore's law? Science **368**(6495), eaam9744 (2020)

2. Tilli, A., Garone, E., Conficoni, C., Cacciari, M., Bosso, A., Bartolini, A.: A two-layer distributed mpc approach to thermal control of multiprocessor systems-on-chip. Control Eng. Pract. (2022). https://doi.org/10.1016/j.conengprac.2022.105099

3. Labs, A.: AWS Graviton 2. https://en.wikichip.org/wiki/annapurna_labs/alpine/alc12b00 (2020)

4. Intel: Raptor Lake. https://en.wikichip.org/wiki/intel/microarchitectures/raptor_lake (2022)

5. AMD: EPYC 7004 Genoa. https://en.wikichip.org/wiki/amd/cores/genoa (2022)

6. Group, T.L.: SiPearl Develops ARM HPC Chip. https://www.linleygroup.com/newsletters/newsletter_detail.php?num=6227 &year=2020 &tag=3 (2020)

7. Cesarini, D., Bartolini, A., Bonfa, P., Cavazzoni, C., Benini, L.: COUNTDOWN: a run-time library for performance-neutral energy saving in MPI applications. IEEE Trans. Comput. **1**, 682–695 (2020)

8. Rotem, E., Naveh, A., Ananthakrishnan, A., Weissmann, E., Rajwan, D.: Power-management architecture of the intel microarchitecture code-named Sandy Bridge. IEEE Micro **32**(2), 20–27 (2012). https://doi.org/10.1109/MM.2012.12

9. Bartolini, A., Cacciari, M., Tilli, A., Benini, L.: Thermal and energy management of high-performance multicores: distributed and self-calibrating model-predictive controller. IEEE Trans. Parallel Distrib. Syst. **24**, 170–183 (2013). https://doi.org/10.1109/TPDS.2012.117

10. Beneventi, F., Bartolini, A., Benini, L.: On-line thermal emulation: how to speed-up your thermal controller design. In: 2013 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), pp. 99–106 (2013). https://doi.org/10.1109/PATMOS.2013.6662161

11. Ltd., A.: Power and performance management using arm SCMI specification. Technical report (2019)

12. LLC, G.: Power management for multiple processor cores (U.S. Patent US8402290B2, 2020)

13. Ripoll, I., Ballester, R.: Period selection for minimal hyper-period in real-time systems (2014)

14. Liu, Z., Zhu, H.: A survey of the research on power management techniques for high-performance systems. Softw. Pract. Exp. **40**, 943–964 (2010). https://doi.org/10.1002/spe.v40:11

15. Rosedahl, T., Broyles, M., Lefurgy, C., Christensen, B., Feng, W.: Power/performance controlling techniques in OpenPOWER. In: Kunkel, J.M., Yokota, R., Taufer, M., Shalf, J. (eds.) High Performance Computing, pp. 275–289. Springer, Cham (2017)

16. Schlager, M., Obermaisser, R., Elmenreich, W.: A framework for hardware-in-the-loop testing of an integrated architecture, vol. 4761, pp. 159–170 (2007). https://doi.org/10.1007/978-3-540-75664-4_16

17. Ottaviano, A., Balas, R., Bambini, G., Bonfanti, C., Benatti, S., Rossi, D., Benini, L., Bartolini, A.: ControlPULP: a RISC-V power controller for HPC processors with parallel control-law computation acceleration. In: Orailoglu, A., Reichenbach, M., Jung, M. (eds.) Embedded Computer Systems: Architectures, Modeling, and Simulation, pp. 120–135. Springer, Cham (2022)

18. Rossi, D., Conti, F., Marongiu, A., Pullini, A., Loi, I., Gautschi, M., Tagliavini, G., Capotondi, A., Flatresse, P., Benini, L.: PULP: a parallel ultra low power platform for next generation IoT applications. In: 2015 IEEE Hot Chips 27 Symposium (HCS), pp. 1–39 (2015)

19. RISC-V: "Smclic" core-local interrupt controller (CLIC) RISC-V privileged architecture extension. https://github.com/riscv/riscv-fast-interrupt/blob/master/clic.adoc

20. Bambini, G., Balas, R., Conficoni, C., Tilli, A., Benini, L., Benatti, S., Bartolini, A.: An open-source scalable thermal and power controller for HPC processors. In: 2020 IEEE 38th International Conference on Computer Design (ICCD), pp. 364–367 (2020)

21. Gunther, S., Deval, A., Burton, T., Kumar, R.: Energy-efficient computing: power management system on the Nehalem family of processors. Intel Technol. J. **14**(3), 50–66 (2010)

22. Schöne, R., Ilsche, T., Bielert, M., Gocht, A., Hackenberg, D.: Energy efficiency features of the Intel Skylake-SP processor and their impact on performance. In: 2019 International Conference on High Performance Computing Simulation (HPCS), pp. 399–406 (2019). https://doi.org/10.1109/HPCS48598.2019.9188239

23. Burd, T., Beck, N., White, S., Paraschou, M., Kalyanasundharam, N., Donley, G., Smith, A., Hewitt, L., Naffziger, S.: "Zeppelin": an SoC for multichip architectures. IEEE J. Solid-State Circuits **54**(1), 133–143 (2019). https://doi.org/10.1109/JSSC.2018.2873584

24. ARM Ltd.: Arm system control and management interface V3.0. ARM Ltd. https://developer.arm.com/documentation/den0056/latest

25. Atienza, D., Del Valle, P.G., Paci, G., Poletti, F., Benini, L., De Micheli, G., Mendias, J.M.: A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. In:

2006 43rd ACM/IEEE Design Automation Conference, pp. 618–623 (2006). https://doi.org/10.1145/1146909.1147068

26. Atienza, D.: Emulation-based transient thermal modeling of 2D/3D systems-on-chip with active cooling. In: 2009 15th International Workshop on Thermal Investigations of ICs and Systems, pp. 50–55 (2009)

27. Brayanov, N., Eichberger, A.: Automation in hardware-in-the-loop units development and integration. In: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 191–197 (2019). https://doi.org/10.1109/QRS-C.2019.00047

28. Tan, Z., Waterman, A., Cook, H., Bird, S., Asanović, K., Patterson, D.: A case for fame: FPGA architecture model execution. In: Proceedings of the 37th Annual International Symposium on Computer Architecture. ISCA '10, pp. 290–301. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1815961.1815999

29. RISC-V: the RISC-V instruction set manual volume II: privileged architecture. RISC-V. https://riscv.org/technical/specifications/

30. Montagna, F., Mach, S., Benatti, S., Garofalo, A., Ottavi, G., Benini, L., Rossi, D., Tagliavini, G.: A low-power transprecision floating-point cluster for efficient near-sensor data analytics. IEEE Trans. Parallel Distrib. Syst. **33**(5), 1038–1053 (2022). https://doi.org/10.1109/TPDS.2021.3101764

31. Rossi, D., Loi, I., Haugou, G., Benini, L.: Ultra-low-latency lightweight DMA for tightly coupled multi-core clusters. In: Proceedings of the 11th ACM Conference on Computing Frontiers. CF '14. Association for Computing Machinery, New York, NY, USA (2014)

32. Kurth, A., Forsberg, B., Benini, L.: HEROv2: full-stack open-source research platform for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **33**(12), 4368–4382 (2022). https://doi.org/10.1109/TPDS.2022.3189390

33. Beneventi, F., Bartolini, A., Tilli, A., Benini, L.: An effective gray-box identification procedure for multicore thermal modeling. IEEE Trans. Comput. **63**(5), 1097–1110 (2014)

34. Bambini, G., Conficoni, C., Tilli, A., Benini, L., Bartolini, A.: Modeling the thermal and power control subsystem in HPC processors. In: 2022 IEEE Conference on Control Technology and Applications (CCTA), pp. 397–402 (2022). https://doi.org/10.1109/CCTA49430.2022.9966082

35. Das, S., Whatmough, P., Bull, D.: Modeling and characterization of the system-level power delivery network for a dual-core ARM Cortex-A57 cluster in 28nm CMOS. In: 2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 146–151 (2015). https://doi.org/10.1109/ISLPED.2015.7273505

36. Bartolini, A., Ficarelli, F., Parisi, E., Beneventi, F., Barchi, F., Gregori, D., Magugliani, F., Cicala, M., Gianfreda, C., Cesarini, D., Acquaviva, A., Benini, L.: Monte Cimone: paving the road for the first generation of RISC-V high-performance computers. arXiv (2022). https://doi.org/10.48550/ARXIV.2205.03725. https://arxiv.org/abs/2205.03725

37. Bienia, C., Kumar, S., Singh, J.P., Li, K.: The PARSEC benchmark suite: characterization and architectural implications. In: Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. PACT '08, pp. 72–81. Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1454115.1454128

38. Müller, M., Whitney, B., Henschel, R., Kumaran, K.: In: Padua, D. (ed.) SPEC Benchmarks, pp. 1886–1893. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-09766-4_370

39. Hennessy, J.L., Patterson, D.A.: Computer Architecture: A Quantitative Approach, 5th edn. Morgan Kaufmann, Amsterdam (2012)

40. Borghesi, A., Burrello, A., Bartolini, A.: ExaMon-X: a predictive maintenance framework for automatic monitoring in industrial IoT systems. IEEE Internet Things J. (2021). https://doi.org/10.1109/JIOT.2021.3125885

Authors and Affiliations

**Alessandro Ottaviano[1] · Robert Balas[1] · Giovanni Bambini[2] ·
Antonio Del Vecchio[2] · Maicol Ciani[2] · Davide Rossi[2] · Luca Benini[1,2] ·
Andrea Bartolini[2]**

✉ Alessandro Ottaviano
  aottaviano@iis.ee.ethz.ch

  Robert Balas
  balasr@iis.ee.ethz.ch

  Giovanni Bambini
  giovanni.bambini2@unibo.it

  Antonio Del Vecchio
  antonio.delvecchio5@unibo.it

  Maicol Ciani
  maicol.ciani@unibo.it

  Davide Rossi
  davide.rossi@unibo.it

  Luca Benini
  lbenini@iis.ee.ethz.ch

  Andrea Bartolini
  a.bartolini@unibo.it

[1] Integrated Systems Laboratory, ETH Zürich, Gloriastrasse 35, 8092 Zurich, Switzerland

[2] DEI, University of Bologna, Viale Del Risorgimento 2, 40136 Bologna, Italy