ControlPULP: A RISC-V Power Controller for HPC Processors with Parallel Control-Law Computation Acceleration

Author(s):
Ottaviano, Alessandro; Balas, Robert; Bambini, Giovanni; Bonfanti, Corrado; Benatti, Simone; Rossi, Davide; Benini, Luca; Bartolini, Andrea

Publication date:
2022

Permanent link:
https://doi.org/10.3929/ethz-b-000580004

Originally published in:
Lecture Notes in Computer Science 13511, https://doi.org/10.1007/978-3-031-15074-6_8

Funding acknowledgement:
101034126 - Pilot using Independent Local & Open Technologies (EC)
101036168 - European Processor Initiative (EPI) SGA2 (EC)

ControlPULP: A RISC-V Power Controller for HPC Processors with Parallel Control-Law Computation Acceleration

Alessandro Ottaviano[1], Robert Balas[1], Giovanni Bambini[2], Corrado Bonfanti[2], Simone Benatti[2], Davide Rossi[2], Luca Benini[1,2], and Andrea Bartolini[2]

[1] ETH Zurich, Switzerland
[2] University of Bologna, Italy

Abstract. High-Performance Computing (HPC) processors are nowadays integrated Cyber-Physical Systems requiring complex and high-performance closed-loop control strategies for efficient power and thermal management. To satisfy high-bandwidth, real-time multi-input multi-output (MIMO) optimal power control requirements, high-end processors integrate on-die Power Controller Systems (PCS). Traditional PCS is based on a simple microcontroller core supported by dedicated interface logic and sequencers. More scalable and flexible PCS architectures are required to support advanced MIMO control algorithms required for managing the ever-increasing number of cores, power states, and process, voltage, temperature (PVT) variability.
In this paper, we present ControlPULP, a complete, open-source HW/SW RISC-V parallel PCS platform consisting of a single-core microcontroller coupled with a scalable multi-core cluster system with a specialized DMA engine and a fast multi-core interrupt controller for parallel acceleration of real-time power management policies. ControlPULP relies on a real-time OS (FreeRTOS) to schedule a Power Control Firmware (PCF) software layer. We evaluate ControlPULP design choices in a cycle-accurate, event-based simulation environment and show the benefits of the proposed multi-core acceleration solution. We demonstrate ControlPULP in a PCS use-case targeting a next-generation 72-cores HPC processor. We show that the multi-core cluster accelerates the PCF achieving 4.9x speedup with respect to single-core execution.

Keywords: RISC-V · HPC Processor · Power and Thermal Control · Scalable · Parallel microcontroller.

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 technological scaling. This trend has made the processing elements at the heart of computing nodes energy, power, and thermally constrained [10]. Modern high-performance processors feature a large number of cores. The most notable

examples are AWS Graviton 2 (64 ARM Neoverse N1 cores) [9], Intel Alder Lake-S Xeon (16 cores, 24 threads) [8], AMD Epyc 7003 Milan (up to 64 Zen 3 cores) [1], SiPearl Rhea Processor (72 ARM Neoverse V1 Zeus cores) [6] and the NVIDIA Grace CPU Superchip (144 ARM Neoverse N2 cores). Their application workload requires a dynamic trade-off between maximum performance (fastest operating point [5]) in CPU-bound execution phases and energy efficiency in memory-bound execution phases (energy-aware CPU [21]). Hence, all modern processors integrate on-die Power Controller Subsystems as dedicated hardware resources co-designed with a Power Control Firmware (PCF) implementing complex MIMO power management policies. Advanced power management involves embedding and interleaving a plurality of activities in the PCS, namely (i) dynamic control of the CPU power consumption with short time constants [16] to prevent thermal hazards and to meet the TDP limit (power capping [11]), (ii) real-time interaction with inputs provided by on-die (Operating System - OS - power management interfaces and on-chip sensors) and off-die (Baseboard Management Controller - BMC -, Voltage Regulator Modules - VRMs -) units and (iii) dynamic power budget allocation between general-purpose (CPUs) and other integrated subsystems, such as graphics processors (GPUs) [21].

Existing on-die PCSs share a common design structure with an integrated single-core microcontroller [3] supported by dedicated hardware state machines [21] or more generic accelerators [18] (Sec. 2). The hardware typically takes advantage of specific software libraries [4,3] to implement the real-time execution environment required to run power management policies under tight timing constraints.

Many-core power management requires fine-grained control of the operating points of the processing elements [12] to meet a given processor power consumption setpoint while minimizing performance penalties. Moreover, the control policy has to provide fast, reactive, and predictable responses to promptly handle the incoming requests from the OS or BMC and prevent thermal hazards. A flexible and scalable way to manage these computationally intensive operations is required to provide a high-quality level of control performance per core and to support more advanced experimental control policies. This scenario suggests the need for a performant and capable PCS architecture optimized for handling a fine-grained, per-core performance state control strategy on a large number of controlled cores within the required timing deadlines. In this work, we address the requirement mentioned above and make 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 [19]. To the best of our knowledge, ControlPULP is the first fully open-source [5] (hardware and software) 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 (Sec. 3). ControlPULP integrates a multi-core cluster with per-core

[3] `https://github.com/ARM-software/SCP-firmware`

[4] `https://github.com/open-power`

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

FPUs for reactive control policy step computation, without the additional complexity of floating-point to fixed-point conversion.
2. The cluster integrates a specialized DMA to accelerate the data transfers from on-chip sensors and off-chip peripherals. It allows data acquisition from 2D strided data access patterns, a crucial capability when reading from private, per-core sensors with equally spaced address mapping (Sec. 4.3).
3. We tailor ControlPULP to meet real-time power management requirements. The architecture achieves low interrupt latency thanks to a platform-level interrupt controller (RISC-V PLIC [17]) tasked to process the global interrupts associated with OS- and BMC- driven commands and a low latency predictable interconnect infrastructure (Sec. 4.3).
4. We demonstrate the end-to-end capabilities of ControlPULP with a case study on the control quality of the PCF, that achieves 6% more precise setpoint tracking than the only openly documented SoA control policy implemented by the IBM on-chip controller (Sec. 4.5).

2 Related Work

There is little publicly available information on commercial state-of-the-art (SoA) PCS architectures. To the best of the authors' knowledge, the main four market-standard PCS are: Intel Power Control Unit (PCU) introduced with Nehalem microprocessor [7], the ARM System Control Processor (SCP) [13], the System Management Units (SMU) integrated with AMD Zeppelin, and the IBM On-Chip Controller (OCC) introduced with Power8 microprocessor.

Intel's PCU is a combination of dedicated hardware state machines and an integrated microcontroller [21]. It provides power management functionalities such as Dynamic Voltage Frequency Scaling (DVFS) through voltage-frequency control states (P-states and C-states), selected by the HW (Hardware-Managed P-States, HWP). The PCU communicates with the processing elements with a specialized link through Power Management Agents (PMAs). Intel's main control loop runs at $500\mu s$ [22].

AMD adopts a multiple power controller design, with one SMU for each CPU tile (group of cores). All 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 [4].

ARM implements two independent PCSs based on the ARM Cortex-M7 microcontroller, SCP and MCP (System and Manageability Control Processor, respectively). 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 [2]. 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 power controller, called OCC, is composed of 5 units: a central PowerPC 405 processor with 768 KiB of dedicated SRAM and four general-purpose

engines for data collection from PVT sensors (GPEs) and performance state and CPU stop functions control (PGPE and SGPE) respectively. IBM's main control loop runs on PowerPC 405 at $250\mu s$ [18]. It uses a Frequency Voting Box 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 PID with a periodicity of $16ms$ that reduces the frequency of each core based on the temperature of the hottest element. Furthermore, similarly to ARM's SCMI standard, IBM's OCC uses a Command Write Attention/Interrupt mechanism to notify the PGPE of an incoming asynchronous command/request to be processed[4], for instance, the desired PState. PGPE arbitrates this information with the Voting Box output from the PowerPC 405 according to a minimum PState policy.

Last, it has been shown [3] that a single-core, RISC-V based microcontroller can execute similar control algorithms with a control loop of $500\mu s$. Nevertheless, the underlying PCS architecture did not support an adequate I/O system infrastructure with on-die and off-die actors, global interrupt lines for OS-based commands dispatching, and a floating-point unit for the PCF's workload routine.

All the SoA power controllers lack the flexibility and scalability of a multicore architecture supported by adequate IO bandwidth and fast interrupt handling hardware, which is the critical innovation provided by ControlPULP.

3 The ControlPULP platform

ControlPULP extends commercial controllers' single-core microcontroller structure, providing the first multi-core RISC-V PCS architecture. In the following, we detail ControlPULP's hardware architecture (Sec. 3.1) and its design trade-offs. To further analyze the benefits of ControlPULP's design choices with a control algorithm use-case (Sec. 4), we also describe the PCF software structure and the surrounding controlled ecosystem in Sec. 3.2.

3.1 System architecture

Fig. 1 provides a block diagram overview of ControlPULP. The top-level subsystem of the design is the Manager Core, a CV32E40P open-source [6] industry-grade processor core and a set of System I/O interfaces to interact with peripherals and memory-mapped devices (Sec. 3.1). The primary micro-controller-like subsystem is also present in a similar form in the SoA designs surveyed in Sec. 2.

Real-time and predictability The following architectural design decisions were taken concerning RAM banking and interrupt processing to make the design more suitable for real-time workloads.

[6] `https://github.com/openhwgroup/cv32e40p`

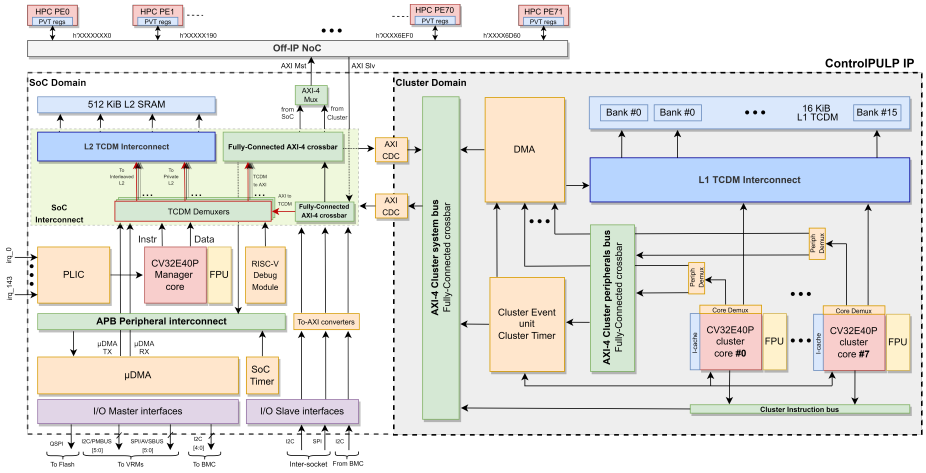

Fig. 1: ControlPULP architecture.

L2 memory banks constant access time The L2 RAM, which is the RAM block connected to the Manager Core 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 PLIC Interrupt controller We integrate a low and constant latency RISC-V interrupt controller (Platform Level Interrupt Controller, PLIC) [17] capable of entering the interrupt handler within 46 cycles (Sec. 4.3). The PLIC can handle up to 144 secure and non-secure global interrupt lines fed to ControlPULP from the mailbox infrastructure. The PLIC multiplexes them in the SoC domain Manager Core as external interrupts. The PLIC is paired with the existing Core Local Interrupt (CLINT) unit in the CV32E40P core, managing software and timer interrupts. The PLIC/CLINT configuration performs hardware-based priority arbitration with vectored interrupts that helps reduce the interrupt response latency (Sec. 4), a crucial property to increase responsiveness on external, agents-driven requests.

Cluster accelerator To meet the computational demands of the control algorithms, especially when scaling to a large number of controlled high-performance cores and improving the control performance, we opt for a flexible programmable accelerator, namely a cluster of RISC-V processors. The cluster consists of a team of CV32E40P cores (workers) tightly coupled to 128 KiB RAM (L1) and a DMA engine.

Multi-core computing system Control algorithms (Sec. 3.2) can be parallelized on the cluster (Sec. 4.2), with a high level of flexibility on the RISC-V cores to improve and update control algorithms. This is in sharp contrast with hard-wired control logic featured in SoA controllers (Sec. 2) which lack flexibility. The Manager Core offloads the control algorithm to the team of workers in the cluster. Each worker has a private instruction cache that copies the instructions to be fetched from L2 and accesses L1 through a single-cycle latency logarithmic interconnect.

In the most straightforward parallelization scheme, a worker computes the control action (Sec. 3.2) for a subset of the controlled cores. The number of workers in the cluster is parametric. In the following, we assume it equals 8, a pretty large configuration, 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. Furthermore, Montagna et al. [15] show that this configuration achieves high performance and reasonable area/energy efficiency on a large number of 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, Sec. 4.3). 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 cores [20].

We tailored 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 PCS also separate the actual computation from data acquisition. For instance, IBM OCC employs general-purpose cores (GPEs) tasked to read the processing elements data and temperatures instead of a data mover engine with a micro-coded programming interface [18]. Our DMA-based solution achieves higher performance than data-mover cores and reduces hardware cost.

System I/O interfaces

Low latency AXI4 external 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 ControlPULP design and guarantee low-latency communication with the controlled system. The AXI slave maps to a region of the SoC domain's L2 SRAM drives the PCS's booting process, and loads the PCF binary into L2 SRAM. The AXI master is the transport layer over which the PCS collects PVT sensors data and power policy target requirements. It dispatches the optimal frequency operating point during the control policy

(Sec. 3.2). We internally routed this channel to initiate data transfers from both the SoC and Cluster domains through the arbitration of fully-connected AXI4 crossbars.

SCMI ControlPULP adopts and implements the ARM standard SCMI protocol to handle external power, performance, and system management requests. SCMI allows an OS kernel that supports SCMI to interact with ControlPULP without needing a bespoke driver directly. Furthermore, the design of the SCMI protocol reflects the industry trend of delegating power and performance to a dedicated subsystem [14]. SCMI involves an interface channel for secure and non-secure communication between a caller (agent, i.e. an HPC processing element) and a callee (platform, i.e. ControlPULP). The latter interprets the messages delivered by the former in a shared memory area (mailbox region, Fig. 2a) and responds according to a specific protocol. The proposed PCS implements a doorbell-based (interrupt-driven) transport mechanism through the PLIC. In our use case with 72 controlled cores, the platform can process up to 144 secure and non-secure interrupt notifications.

High latency peripherals ControlPULP integrates a peripheral subsystem in the SoC domain inherited from the PULP design, where an I/O data engine unit (μDMA) allows autonomous communication between off-die elements and the L2 SRAM. In this work, we upgrade the peripheral subsystem to handle off-die communication services through industry-standard power management interfaces. The PCS integrates 6 AVSBUS and PMBUS interfaces to VRMs and 1 QSPI to external non-volatile memory, while 5 I2C master/slave interfaces manage the communication with the BMC. The Power Management Bus (PMBUS) and Advanced Voltage Bus (AVSBUS) bus protocols extend I2C and SPI respectively to provide digital monitoring of voltage and power rails, preserving optimal speed/power consumption trade-off.

3.2 Power Control Firmware

The PCF executes the thermal and power control functions and manages on-die and off-die communications and data transfers. FreeRTOS, an industry-grade, lightweight, and open-source operating system for microcontrollers, serves as the basis for real-time priority-driven scheduling with static tasks priorities and preemption.

The control routine consists of two periodic tasks characterized by multiple harmonic frequencies: the **Fast Power Control Task (FPCT)** (8 kHz) and the **Periodic Control Task (PCT)** (2 kHz). Splitting the control routine into two tasks grants more fine-grained scheduling and helps meet different performance requirements and sensors-update frequencies. Power has faster changes due to instruction-level variation of the effective capacitance of the computing unit, while temperature variations are slower. The control has to handle these widely split time scales. Furthermore, power sensors (VRMs) generally update more frequently than temperature sensors.

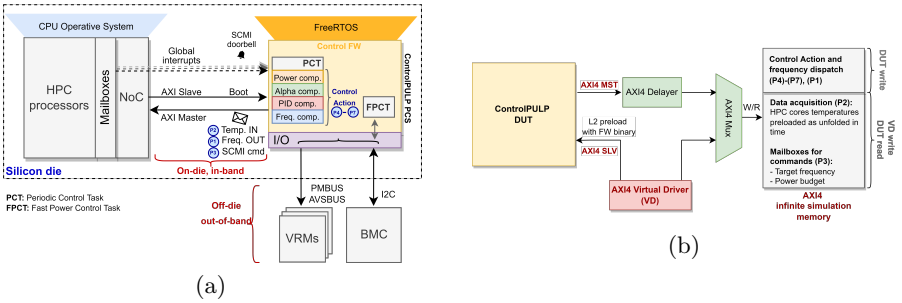

Fig. 2: **(a)** PCF inputs and outputs interactions. **(b)** ControlPULP RTL testbench simulation environment.

The PCT is the main control task. It receives the desired operating point (clock frequency) for each processing element and controls it to meet the physical and imposed constraints of the system. It executes a two-layer control strategy [3] consisting of a Power Dispatching Layer and a Thermal Regulator Layer. The PCT control step n consists of 7 phases: **(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 for step $n+1$; **(P3)** Obtain commands and information on the constraints (target frequency, power budget) from the OS and 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 [3] when the total power exceeds the power budget constraint; **(P6)** Further reduce the power of each core through PIDs computation when the temperature at phase (P2) from step $n-1$ exceeds the threshold; **(P7)** Compute and dispatch a frequency to apply at (P1) in step $n+1$. The transient data computed in these phases are saved for telemetry purposes.

The FPCT tackles the changes in the power consumption of the system. It periodically reads the power consumption of the voltage rails from the VRMs, programs micro-architectural power/instruction throughput capping interfaces (if supported by the processing element), and modifies the power budget threshold of the PCT as requested by the BMC.

Fig. 2a depicts the real-time inputs and outputs the PCF has to interact with. '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 registers data acquisition. 'Off-die' indicates VRMs communication and BMC requests through out-of-band services. Last, we name 'Control Action' the computational body of the PCF execution (P4)-(P7).

4 Experimental Results

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

- We assess and break down ControlPULP's post-synthesis area and determine the minimum area overhead ($< 1\%$) with respect to a modern HPC processor chip (Sec. 4.1).
- We evaluate ControlPULP with a testbench running in a cycle-accurate simulation environment, depicted in Fig. 2b. The testbench does not emulate the HPC processor at the RTL level. Instead, we model the closed-loop with a shared memory region between the PCS platform — the device under test — and the system under control. The real-time temperature and telemetry information from the HPC processor are computed beforehand in software (MATLAB Simulink) and stored in the simulation memory as unfolded in time. Furthermore, we model the interconnect network (Fig. 2a) latency by introducing a programmable latency into the AXI4 ports.
- In the described test scenario, we first study the parallelization of the Control Action (P4)-(P7) on the cluster (Sec. 4.2). We then characterize in-band transfers, namely strided DMA accesses for data acquisition from PVT registers and low interrupt latency with SCMI command processing (Sec. 4.3). Finally, we show the overall performance improvement when accelerating control tasks in the cluster compared to single-core (Sec. 4.4).
- We evaluate the PCF control policy quality in a pure software-based simulation using MATLAB Simulink. We show that the PCF compares favorably against the most well documented and open SoA industrial solution on the market, the IBM OCC (Sec. 4.5).

4.1 Area evaluation

We synthesize ControlPULP in GlobalFoundries 22FDX FD-SOI technology using Synopsys Design Compiler 2019.12. For this technology, one gate equivalent (GE) equals 0.199 μm^2. The design has an overall area of 9.1 MGE when imposing a system clock frequency of 500 MHz. As from the area breakdown shown in table 1, the cluster accelerator accounts for about 32% of the design.

The target controlled system die area is assumed comparable to other commercials, multi-core (> 64) server-class processors, such as [9] (about 457 mm^2). By scaling the gate-equivalent count of the HPC CPU die in the same technology node of this work, ControlPULP would still represent less than 1% of the available die area[7]. 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.

4.2 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 [3])

[7]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.

Table 1: ControlPULP post-synthesis area breakdown on GF22FDX technology.

Unit	Area [mm^2]	Area [kGE]	Percentage [%]
Cluster unit	0.467	2336.7	25.5
SoC unit	0.135	675.9	7.39
L1 SRAM	0.119	595.7	6.51
L2 SRAM	1.108	5542.1	60.6
Total	**1.830**	**9150.3**	**100**

to evaluate each computational phase fairly. Each cluster core is responsible for a subset of the controlled processing elements. The parallelisation is implemented as a fork-join process where the workload is statically distributed among the workers. In ControlPULP, the construct is implemented by means of a per-worker `thread_id` $\in [0 : N_{workers} - 1]$, and an equally distributed `chunk_size` where `chunk_size` $= \frac{N_{ctrl\_cores}}{N_{workers}}$. 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 Figs. 3a and 3b respectively. The 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 processing elements, 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 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 [3] 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. The overhead from synchronization and single-core reduction accounts for 112 and 24 clock cycles, respectively.

As it can be seen from the above analysis, the increased parallel compute capability to handle the workload of the control routine, paired with the general

Table 2: Interrupt latency from interrupt edge to the first instruction in the interrupt handler in number of cycles.

Location	**Increment [cycles]**	**Sum [cycles]**
PLIC input to output	2	2
CLINT input to core	7	9
Jump in vector table to PLIC handler	2	11
Save caller save regs (addi + 15 regs)	17	28
Claim PLIC interrupt (read id)	8	36
Compute and load PLIC handler address	8	44
Jump to PLIC handler address	2	46
Summary	-	**46**

purpose nature of the accelerator, enable us to (i) improve the control performances paving the way to more advanced algorithms and (ii) be fully flexible when designing the control algorithm.

4.3 In-band Services

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 AXI 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 [20]. 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 Sec. 4.2, we use synchronization barriers to coordinate the eight cores. Fig. 3c 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 processing elements and the information needed from them (P, V, T, i.e. ≥ 3 lower bound). Fig. 3c shows that the best DMA-based transfers assuming 1000 PVT registers (2-D) are 5.3x faster than single-core direct data gathering.

SCMI An interrupt-driven (doorbell) transport regulates the communication in the agent-platform direction — phase (P3) of the Periodic Control Task. Table 2 gives an overview of the overall PLIC interrupt latency measured as the number of cycles from the triggering edge in the PLIC to the ISR Handler's first instruction. The RTL testbench environment (Fig. 2b) emulates the shared mailboxes.

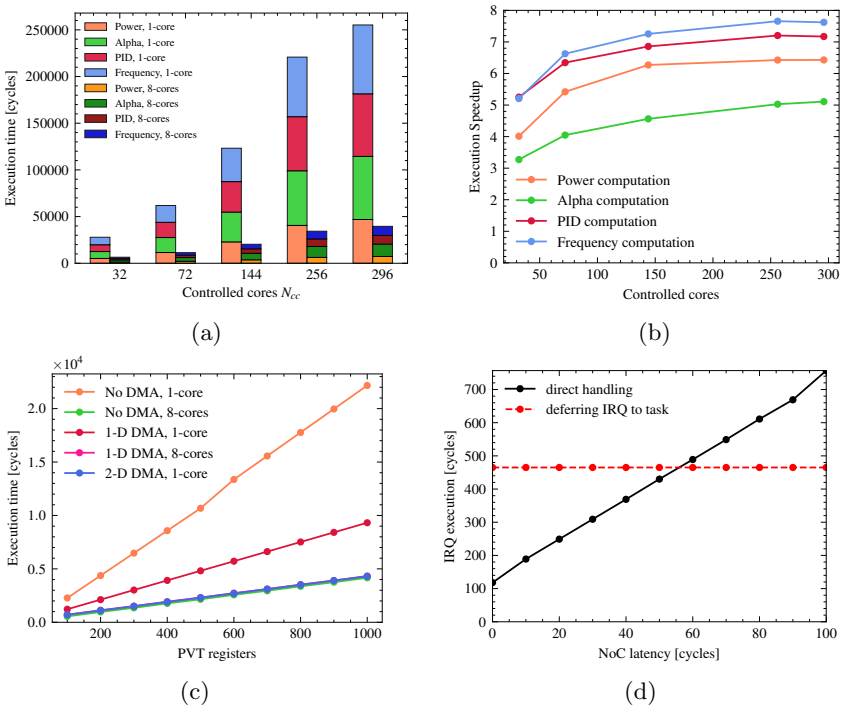

Fig. 3: **(a)-(b)** Firmware Control Action, execution time, and speedup comparison between single-core (SoC domain) and 8-cores (Cluster domain). **(c)** In-band data acquisition from simulated PVT registers, execution time without and with DMA in 1-D and 2-D configurations. **(d)** Execution time in the interrupt handler from the interrupt edge to its completion with a basic SCMI message at varying interconnect network access latency to the mailbox.

Each global interrupt line is paired with a mailbox that can host one command message and is associated with a unique agent over an SCMI channel. The minimum size of an SCMI message (header and payload) is 32B when employing a 4B payload for platform response according to standard specifications [2]. Figure 3d shows the execution time of an SCMI command decoding and response (Base Protocol, `protocol_id = 0x10`, `message_id = 0x0` [2]) when an external simulated driver rings a doorbell to the PCS. In the experiment, we emulate the latency of the interconnect network between ControlPULP and the mailbox location in the die (Figs. 2a) as described in Fig. 2b. The latency has a large impact on the load/store access times, thus the time spent in the ISR, which grows with the interconnect delay size. We address the issue using the FreeRTOS timer API `xTimerPendFunctionCallFromISR()` to defer pending interrupts and keep the ISR time short and insensitive to the CPU interconnect network delay (Fig. 3d). From Fig. 3d, we see that deferring interrupt handling to a task is preferable over direct handling, as it is network-latency insensitive and faster for realistic NoC latencies larger than 50 cycles. Other existing solutions, such

Table 3: Execution time T of a PCT 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)	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**

as ARM SCP firmware, propose a bespoke Deferred Response Architecture [3] 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.

4.4 System-level PCF step evaluation

The standalone evaluations of ControlPULP's architectural features from the previous sections need to be finalized with the overall Periodic Control Task step cycle count comparison between accelerator enhanced and single-core configurations, reported in Table 3 in the case of 72 controlled cores. Table 3 shows a breakdown of the required actions. The total execution time computation differs in the two domains. In the single-core case, we execute sequentially with less overhead ($T_{single} = \tau_0 + \tau_1$). 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, which is essential to keep data telemetry between SoC and cluster domains during the PCT.

Overall, multi-core execution achieves a 4.9x speedup over a single-core configuration. This helps reduce the PCF periodic control policy hyper-period (the least common multiple of the control tasks' periods) [16] and increase the available computation time within the hyper-period, respectively.

4.5 Control-level PCF step evaluation

Control comparison with SoA solutions The control quality of the PCF step is tightly coupled with the architecture of the system to be controlled. Benchmarks comparisons concern the overall HPC chip performance and are not focused on the controller alone. The only commercial solution available for cross-benchmarking is the IBM OCC (Sec. 2).To enable a meaningful comparison, we

use MATLAB Simulink to model (i) the HPC die as control target and (ii) the IBM's control action excluding few architecture-specific features, and the two-layer PCF control [3] executed by ControlPULP. The PID-like coefficients of the IBM control are adapted for the HPC chip model. We assume a constant voltage of $0.75V$ and neither overhead nor delays in the PLLs and VRMs operating point transitions. The simulation runs for $2s$. The HPC chip model consists of a 9-cores tile. Each core executes diverse synthetic[8] workloads: Core 1/Core 3 and Core2/Core 4 execute maximum power (`MAX`) and low power (`LOW`) instructions, respectively. Core 5, 6, and 9 execute heterogeneous mixed instruction (`MIX`), while Core 7 and Core 8 are exposed to sharp instruction types switching (`FAST`) to stress the power limiter and the shorter timing constants of the temperature response. The DVFS target frequency for each core is kept constant at the maximum frequency, while the power budget is changed five times during the simulation to stress all the elements of the control action.

First, we show that a controller with a multi-core cluster able to deliver higher computational power is beneficial for 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. The comparison highlights the positive effect of having fine-grained control made possible by the cluster. The performance, measured as the number of retired instructions, is shown in Figure 4. Using only one temperature for the whole tile results in an average performance reduction per core of 5.55%, while 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. As from Sec. 2, the IBM control policy considers the maximum temperature among the processing elements 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. On the other end, ControlPULP enables fine-grained frequency reduction on a per-core temperature granularity.

Last, we compare the PCF control and the IBM control version with per-core temperature PIDs. The comparison is used to show the validity of the considered PCF control as well as how the accelerator allows fine-grain control decisions by favoring the more demanding workloads. In the latter scenario, the PCF shows a performance increase in executed instructions that ranges from +2.75% to +4.97%. This holds true for cores with mixed instructions (+0.1% to +6.08%) as well, while cores involved in less demanding workloads witness a decrease between −2.72% and −3.76%. Thus, the modified policy with per-core temperature PID calculation, enabled by the parallel cluster, can selectively boost the instructions retired by the critical cores, achieving a higher application performance on the HPC chip while still meeting the thermal cap.

[8] In order to assess the controller, the evaluation of a real workload is out of scope for this work as it requires more complex co-simulation setup. We refer to synthetic workloads that cover relevant corner cases for the ControlPULP.

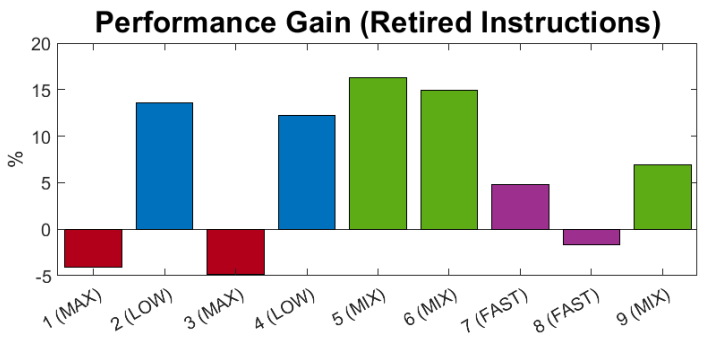

Fig. 4: Performance analysis. Comparison between the original IBM OCC firmware and modified IBM OCC firmware with per-core temperature PID for frequency reduction.

5 Conclusion

In this paper, we presented ControlPULP, a complete RISC-V Power Control System for HPC processors that exploits multi-core capabilities to accelerate control algorithms and features specialized DMA and fast interrupt handling and synchronization. This allows us to apply more fine-grained control policies resulting in better control performance. With the proposed architecture, a control policy step executes 4.9x faster than in single-core configuration. ControlPULP enables the implementation of more complex control algorithms capable of dispatching fine-grained frequency targets with better accuracy.

6 Acknowledgment

The study has been conducted in the context of EU H2020-JTI-EuroHPC-2019-1 project REGALE (g.n. 956560), EuroHPC EU PILOT project (g.a. 101034126), EU Pilot for exascale EuroHPC EUPEX (g. a. 101033975), and European Processor Initiative (EPI) SGA2 (g.a. 101036168).

References

1. AMD: Epyc 7003 Milan. `https://en.wikichip.org/wiki/amd/epyc` (2021)
2. ARM Ltd.: Arm System Control and Management Interface v3.0, `https://developer.arm.com/documentation/den0056/latest`
3. 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)
4. 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 Journal of Solid-State Circuits **54**(1), 133–143 (2019). https://doi.org/10.1109/JSSC.2018.2873584

5. Cesarini, D., Bartolini, A., Bonfa, P., Cavazzoni, C., Benini, L.: Countdown: a run-time library for performance-neutral energy saving in mpi applications. IEEE Transactions on Computers pp. 1–1 (2020)
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. Gunther, S., Deval, A., Burton, T., Kumar, R.: Energy-efficient computing: Power management system on the nehalem family of processors. Intel Technology Journal **14**(3), 50–66 (2010)
8. Intel: Alder Lake. https://en.wikichip.org/wiki/intel/microarchitectures/alder_lake#Codenames (2021)
9. Labs, A.: AWS Graviton 2. https://en.wikichip.org/wiki/annapurna_labs/alpine/alc12b00 (2020)
10. 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) (2020)
11. Liu, Z., Zhu, H.: A survey of the research on power management techniques for high-performance systems. Softw., Pract. Exper. **40**, 943–964 (10 2010). https://doi.org/10.1002/spe.v40:11
12. LLC, G.: Power management for multiple processor cores (US Patent US8402290B2, Dec 2020)
13. Ltd., A.: System control processor firmware, https://developer.arm.com/tools-and-software/open-source-software/firmware/scp-firmware
14. Ltd., A.: Power and Performance Management using Arm SCMI Specification. Tech. rep. (8 2019)
15. 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 Transactions on Parallel and Distributed Systems **33**(5), 1038–1053 (2022). https://doi.org/10.1109/TPDS.2021.3101764
16. Ripoll, I., Ballester, R.: Period selection for minimal hyper-period in real-time systems (2014)
17. RISC-V: RISC-V Platform Level Interrupt Controller, https://github.com/riscv/riscv-plic-spec/blob/master/riscv-plic.adoc
18. 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 International Publishing, Cham (2017)
19. 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)
20. 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)
21. 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
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)