ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

MASTERING POWER CONTROL IN HPC CPUS: A JOURNEY THROUGH MODELING, ALGORITHMS, AND HARDWARE INSIGHTS

Presentata da: Giovanni Bambini

Coordinatore Dottorato

Davide Dardari

Supervisore

Andrea Bartolini

Co-supervisore

Luca Benini

Esame finale anno 2025

UNIVERSITY OF BOLOGNA

Abstract

Ingegneria Elettronica, Telecomunicazioni e Tecnologie dell'Informazione
DEI

Doctor of Electronic, Telecommunications, and Information Technologies Engineering

**Mastering Power Control in HPC CPUs:
A Journey through Modeling, Algorithms, and Hardware Insights**

by Giovanni BAMBINI

High Performance Computing (HPC) has rapidly evolved to meet the increasing computational demands of data-intensive fields such as climate modeling, artificial intelligence, and physics research. This growth is driven by the demand for massive computational power that spans various demographics, including researchers, industry professionals, and governments, arriving at end-users with the growth of Large Language Models (LLMs). Emerging trends in HPC, including many-core and heterogeneous architectures, present significant complexity, especially as they adopt advanced chiplet-based designs with specialized accelerators. These innovations introduce challenges that necessitate sophisticated control strategies to manage power and thermal dynamics effectively.

The open-source RISC-V ISA has spurred the entry of new players into this market segment. However, despite advances in hardware design, there remains a noticeable gap in research concerning on-chip power and thermal control strategies. Existing efforts have largely focused on high-level, software-based control mechanisms at the operating system or application level, leaving low-level control methods underexplored.

This thesis addresses this gap by developing and evaluating advanced low-level control algorithms for power and thermal management in HPC environments. It introduces a comprehensive modeling framework that captures essential system dynamics and highlights the unique challenges of low-level control, such as leakage power management, actuator non-idealities, and coupling constraints. The proposed control strategies, including fuzzy-inspired and Model Predictive Control (MPC) approaches, are validated using a Hardware-in-the-Loop (HIL) testing platform to demonstrate their effectiveness in real-time scenarios.

Results indicate that these advanced controllers significantly enhance thermal regulation, minimize performance degradation, and achieve superior energy efficiency. The thesis concludes by outlining future research directions, such as integrating machine learning for predictive control and exploring distributed control frameworks to further optimize HPC system performance.

Acknowledgements

This thesis would not have been possible without the guidance, support, and encouragement of many individuals and institutions, to whom I am deeply grateful.

First and foremost, I would like to express my sincere gratitude to my primary supervisor, Prof. Andrea Bartolini, for his mentorship, genuine kindness, and insightful advice throughout the entirety of this work. His expertise and continuous support have been fundamental in shaping the direction of this research. I am also immensely thankful to Prof. Luca Benini for his role as co-supervisor, providing critical insights and fostering an environment that encouraged both scientific rigor and intellectual curiosity.

A special acknowledgment goes to Prof. Christian Conficoni for his competence in control and model theory, which greatly contributed to key aspects of this research. I am also grateful to Dr. Alessandro Ottaviano, a colleague and a friend throughout this journey, for his continuous support, stimulating discussions, and the many shared challenges and successes along the way. Additionally, I would like to thank Antonio Del Vecchio for his collaboration on several projects.

During my research abroad at ETH Zürich, I had the privilege of working in an inspiring academic environment, and I extend my sincere appreciation to Irina Rau for her kind support and assistance during my time there.

This work has been supported by multiple research initiatives, and I gratefully acknowledge the funding received from the EU H2020-JTI-EuroHPC-2019-1 project REGALE (g.n. 956560), the EuroHPC EU PILOT project (g.a. 101034126), the EU Pilot for Exascale EuroHPC EUPEX (g.a. 101033975), and the European Processor Initiative (EPI) SGA2 (g.a. 101036168).

Beyond the research work, my PhD journey has been enriched by an exceptional group of colleagues and friends. I am sincerely grateful to all my colleagues at the NeuroLab for creating a truly unique and lively research environment—one that often felt like a scene from The Big Bang Theory. The shared lunches, the legendary "Fraido" sushi Fridays, and the moments of laughter that brightened our "gloomy and crumbling" lab made this experience etched in my memory. A special thanks to Davide Nadalini, Benedetta Mazzoni, Luca Bompani, Marcello Zanghieri, Alberto Dequino, Giuseppe Tagliavini, Mattia Orlandi, Lorenzo Lamberti, Seyed Ahmad Mirsalari, Francesco Conti, Manuele Rusci, and Davide Rossi for making the lab a special place.

Finally, I am profoundly grateful for the unwavering support of my close friends, partner, and family. Their constant encouragement, patience, and belief in me, even during the most challenging moments, have been invaluable. In particular, Laura's presence and support through the most difficult moments meant more than words can express. To all the incredible friendships formed in Bologna that have accompanied me throughout this journey—thank you for making these three years the best of my life.

Contents

List of Figures

List of Tables

Abbreviations

αWHA	α-Weighted Heuristic Algorithm.
ACPI	Advanced Configuration and Power Interface.
AI	Artificial Intelligence.
AP	Application-class Processor.
API	Application Programming Interface.
AVSBus	Adaptive Voltage Scaling Bus.
AVX	Advanced Vector Extensions.
BA	Baseline Algorithm.
BMC	Board Management Controller.
CLIC	Core-Local Interrupt Controller.
CLINT	Core-Local Interruptor.
CPL	Cycles per Level.
CPU	Central Processing Unit.
DFS	Dynamic Frequency Scaling.
DLDO	Digital Low-Dropout Regulator.
DMA	Direct Memory Access.
DPTM	Dynamic Power and Thermal Management.
DRAM	Dynamic Random Access Memory.
DT	Dispatch Throttling.
DVFS	Dynamic Voltage and Frequency Scaling.
DVS	Dynamic Voltage Scaling.
EBA	Enhanced Baseline Algorithm.
EMA	Exponential Moving Average.
EPAC	European Processor Accelerator.
EPI	European Project Initiative.
FCA	Fuzzy-inspired Iterative Control Algorithm.
FCT	Fast Control Task.
FET	Field Effect Transistor.
FIVR	Fully Integrated Voltage Regulator.
FPGA	Field-Programmable Gate Array.
FPU	Floating Point Unit.
GLC	Global-Level Controller.
GPP	General-Purpose Processor.
GPU	Graphic Processing Unit.

HBM	High-Bandwidth Memory.
HIL	Hardware in the Loop.
HLC	High-Level Controller.
HPC	High-Performance Computing.
IoT	Internet of Things.
IP	intellectual property.
IPC	Instructions per Cycle.
ISA	Instruction Set Architecture.
ISR	Instruction Service Routine.
LB-MPC	Linear Baseline MPC.
LDC	Local Controller for Distributed Control.
LE-MPC	Linear Enhanced MPC.
lin-MPC	Linearized MPC.
LLC	Low-Level Controller.
LLM	Large Language Model.
LRP	Light Reference Platform.
LUT	look-up table.
MCP	Manageability Control Processor.
MCTP	Management Component Transport Protocol.
MIL	Model in the Loop.
MIMO	Multiple-Input Multiple-Output.
ML	Machine Learning.
MPC	Model Predictive Control.
MQTT	Message Queuing Telemetry Transport.
NoC	Network on Chips.
OCC	On-Chip Controller.
ODE	Ordinary Differential Equation.
OS	Operating System.
PCB	Printed Circuit Board.
PCI	Peripheral Component Interconnect.
PCT	Periodic Control Task.
PDE	Partial Differential Equation.
PDN	Power Delivery Network.
PE	Processing Element.
PID	Proportional Integral Derivative.
PL	Programmable Logic.
PLDM	Platform Level Data Model.
PLL	Phase-Locked Loop.
PMBus	Power Management Bus.
PS	Processing System.
PVT	Process, Voltage, Temperature.
QP	Quadratic Programming.

RAPL	Running Average Power Limit.
RISC	Reduced Instruction Set Computer.
RTOS	Real-Time OS.
SCMI	System Control and Management Interface.
SCP	System Control Processor.
SCT	Slow Control Task.
SoC	System-on-Chip.
TAP	Timer Application Periodicity.
TDP	Thermal Design Power.
TEP	Target Efficient Power.
TIM	Thermal Interface Material.
TP-MPC	Classic Thermal-Power MPC.
TPC	Thermal and Power Controller.
VBA	Voting Box Algorithm.
VRM	Voltage Regulator Module.

List of Symbols

i,j,k	Index variables	
i	PE index	
j	Voltage Domain index	
k	Time/Iteration index	
t_k	Time instant	
t_s	Execution Time Interval	[s]
t_{th}	Time Discretization of the Thermal Model	[s]
n_c	Number of Cores/PEs	
n_d	Number of (Voltage) Domains	
$\mathcal{D}$	(Voltage) Domain	
T	Temperature	[°C]
T_{L}	Thermal Limit	[°C]
T_{Si}	Silicon Temperature	[°C]
T_{Cu}	Copper/IHS Temperature	[°C]
P	Power	[W]
P_{stat} / P_{s}	Static Power (or Leakage Power)	[W]
P_{dyn}	Dynamic Power	[W]
P_{T} / P^*	Target Power	[W]
P_{M}	"Power-like" MPC control input	[$\text{V}^2\,\text{s}^{-1}$]
P_{B}	Power Budget	[W]
$P_{\mathcal{D}_j}$	Power Budget of the j-th Domain	[W]
P_{est}	Estimated Power (consumption)	[W]
P_{Mes}	Measured Power (consumption)	[W]
ΔP_{C}	Excessive Allocated Power	[W]
F	Frequency	[Hz]
F_{T}	Target Frequency	[Hz]
F_a	Applied Frequency	[Hz]
F_{MA}	Moving Average Frequency	[Hz]
$F_{V-\text{MA}}$	Estimated MA Frequency for Voltage computation	[Hz]
V	Voltage	[V]
V_a	Applied Voltage	[V]
$V_{\mathcal{D}_j}$	Voltage of the j-th Domain	[V]
ω	Workload, Instruction characterization	[−] / [%]
C_{eff}	Effective Capacitance	[F]
h / h_{est}	True / Estimated Power Model	
η	Inverse Power Mapping (h^{-1})	
f_V	Voltage-Frequency TEP function: $V = f_V(F)$	
$g_{\mathcal{D}}$	Voltage Coupling function: $V_{\mathcal{D}} = g_{\mathcal{D}}(V_j)$	
λ, α	Coefficients	
$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$	Linear State-Space Matrixes	

Chapter 1

Introduction

High-Performance Computing (HPC) has become a cornerstone of modern technological progress [82, 84, 105], powering applications in fields ranging from climate modeling [125] and physics simulations [127], to Artificial Intelligence (AI) [74], big data analytics [71], and many more. Its rapid growth is fueled by the increasingly powerful computational resources requirements to tackle these complex, data-intensive tasks. This demand has further intensified in recent years, driven by the rise in popularity of Large Language Models (LLMs) [53], which rely heavily on HPC for training large-scale models, and by global challenges such as the COVID-19 pandemic [139]. At the same time, the international push toward sustainability, accentuated by the global energy crisis and climate change concerns, has highlighted the need for more efficient and energy-conscious computing solutions [67, 121]. In this context, innovations in HPC are increasingly focused on optimizing power efficiency while maintaining cutting-edge performance.

For years, a handful of established companies dominated the HPC chip market [153]. Recently, however, the landscape has shifted with the emergence of new players [86, 21], many encouraged by the openness of the RISC-V Instruction Set Architecture (ISA), which stimulated innovation and enabled a range of companies to pursue processor designs, with some specifically targeting the HPC sector [37]. However, HPC chip design extends beyond just the hardware architecture of computing components; it also involves management and control functions, such as thermal and power regulation. While the hardware delivers the core computational capabilities, robust thermal and power management is important to sustain efficient and reliable operational performance [49].

Although interest in HPC chip design has surged, research in thermal and power management within this domain has largely focused on high-level control methods at the application or Operating System (OS) level, with comparatively less emphasis on direct chip-level regulation [16]. The author of this work conjectures this is mainly due to the industrial secrets covering and restricting access to the HPC chips, their details and technologies, making it difficult to study and modify chip-level functionalities. Consequently, research on Low-Level Controllers (LLCs)—hardware on-chip controllers that directly manage core operations, sensors, and actuators—has stagnated, with most progress in this area dating back over a decade (4.1).

This thesis aims to address that gap by exploring the current state of low-level control strategies in HPC chips, analyzing the associated challenges, and laying the groundwork for future research. The goal is to not only advance the understanding of HPC chip thermal and power control, but also to spark new academic interest in this area, fundamental for next-generation HPC systems.

The favorable circumstances that enabled this research were made possible through the participation in the European Project Initiative (EPI) [81], a project dedicated to regaining technological independence in Europe by developing a competitive

HPC chip on European soil [28]. This setting provided the opportunity to actively participate in architectural design meetings and gain an in-depth understanding of modern chip design, requirements, and technical challenges. In this work, the insights and lessons learned throughout this process will be described, with a particular focus on the power management and thermal control of the chip's cores.

1.1 The European Processor Initiative (EPI) Project

The European Project Initiative (EPI) is a strategic project aimed at reducing Europe's dependency on non-European technologies in the field of HPC. Launched in 2018 as part of the European Union's Horizon 2020 program, EPI seeks to secure Europe's technological sovereignty by designing an energy-efficient high-performance processor tailored to the needs of future European HPC systems, AI applications, and automotive technologies. The initiative is a direct response to Europe's reliance on foreign technology for critical computing infrastructure, which poses potential risks for both national security and competitiveness in a data-driven global economy [28].

One of the unique features of the EPI is its focus on open-source technologies, specifically the adoption of the RISC-V architecture. This decision allows Europe to build customizable Central Processing Units (CPUs) without relying on proprietary solutions from non-European entities. The core of the EPI project is the development of the Rhea HPC General-Purpose Processor (GPP), alongside the creation of the European Processor Accelerator (EPAC) optimized for HPC vector workloads.

The consortium leading the EPI project includes major European industrial players, research centers, and universities, such as Atos, STMicroelectronics, the Barcelona and the CINECA Supercomputing Centers, and the Universities of Zürich, Pisa, and Bologna.

1.2 RISC-V Instruction Set

The Instruction Set Architecture (ISA) serves as the foundational interface between hardware and software in any computing system. It defines the set of instructions a processor can execute, including data handling, memory access, and control flow commands [123].

RISC-V was born out of research at the University of California, Berkeley, in 2010, as part of a long history of research into Reduced Instruction Set Computer (RISC) architectures [11]. The original goal was to create a simplified, efficient, and open-source ISA that could avoid the complexities and proprietary constraints found in older architectures like x86 and ARM. Its open-source nature means that anyone can use, modify, and extend it without licensing fees, making it attractive for academic research, startups, and new companies. Since its creation, RISC-V has rapidly gained popularity and has become a widely adopted standard across industries, including HPC and embedded systems.

In HPC, RISC-V's openness supports the development of specialized processors fine-tuned for scientific tasks, and recompiling programs for a new ISA is generally straightforward. By contrast, older ISAs like x86-64 face the burden of backward compatibility, with legacy instructions adding complexity to chip design and power consumption, limiting their efficiency and scalability in modern applications. RISC-V's minimalist, modular approach avoids these issues, enabling energy-efficient, customizable processors that are easier to design and better suited for a wide range of tasks.

RISC-V is now seen as a key driver of innovation in processor design, and it has gained widespread adoption in a variety of sectors. The ability to customize processors for specific tasks, along with the growing ecosystem of tools and software, has made RISC-V a preferred choice for both academia and industry and accelerated its adoption [65].

1.3 Current-era Chips Design

The design of modern HPC chips has undergone a significant transformation in the last twenty years [12]. Traditionally, CPUs followed a monolithic, single-core design philosophy, where performance improvements were largely driven by increasing clock speeds and shrinking transistor sizes. However, as we approached the physical limits of silicon-based technology—encountering stability issues and voltage limitation—the industry had to shift its focus to multicore architectures to keep increasing the processor performance at a constant rate as described by Moore's law. Today's cutting-edge HPC chips integrate large number of cores, earning them the classification of "many-cores architectures". Performance improvement has also relied on heterogeneous architecture characteristics that integrate general-purpose cores with domain-specific accelerators such as tensor cores, Advanced Vector Extensions (AVX), and Field-Programmable Gate Array (FPGA) [80].

As processor designs incorporate increasing numbers of cores, traditional single-die architectures have faced limitations in scalability and manufacturing efficiency. This challenge has driven a shift toward chiplet architectures [104], where various parts of the processor are manufactured as separate, smaller dies, known as chiplets, and then interconnected on a larger substrate. Chiplet-based design marks a significant advancement in modern chip technology, enabling higher production yields and reduced defect rates, as smaller dies are generally easier to fabricate with fewer errors. Additionally, this modular approach supports the integration of heterogeneous IP blocks, allowing designers to combine diverse cores, accelerators, and memory technologies within a single chip. With this flexibility, processors can be tailored to specific workloads by optimizing individual chiplets with different transistor technologies, enhancing performance and energy efficiency [90].

Despite these advancements, the industry is now facing the so-called "dark silicon" problem, a direct consequence of shrinking transistor technology and increasing power densities in modern multicore processors. As chips integrate more and more cores, it becomes infeasible to operate all of them at their maximum power simultaneously without causing thermal runaway or exceeding the chip's Thermal Design Power (TDP) limits. This results in sections of the chip remaining underutilized (thus "dark") to prevent overheating. The dark silicon phenomenon highlights the thermal and power management challenges associated with modern HPC chips [88]. In this context, architectural specialization [169] plays an essential role, allowing HPC systems to handle diverse tasks with high efficiency.

However, as the number of cores and the power density of specialized accelerators increases, so does the challenge of controlling them. Each core and accelerator introduces unique power, thermal, and performance characteristics, adding complexity to the controller. The modularity and heterogeneity of chiplet-based designs is confronting with the requirement of sophisticated control structures needed to manage heat dissipation along with ensuring efficient and reliable power distribution [16].

The emerging trend in chip design of vertical stacking transistors, known as 3D integration, presents itself with a new set of thermal management challenges.

Although fully stacked 3D chips remain experimental, two-layer designs with stacked L3 cache over compute cores have already been incorporated into consumer CPUs [1, 62], particularly in gaming applications where they achieve leading performance [7]. However, optimally managing the thermal dynamics of these vertically integrated layers, where power and heat rapidly fluctuate across closely packed transistors, is essential for ensuring consistent performance and reliability, and is set to become one of the challenges of the coming years [44].

The control challenges of modern HPC chips are further exacerbated by their fastly-varying dynamic workloads. Computational demands may change rapidly and unpredictably and cores may run vastly different tasks, leading to spatial and temporal variations in power consumption and heat generation. For example, a compute-heavy task on one core may produce significantly more heat than a memory-bound task running on a neighboring core, creating hot spots that, if not managed correctly, could lead to performance throttling or even hardware failure. In addition, with shrinking geometries and increasingly dense transistor layouts, the thermal time constants of these chips have decreased, meaning that heat can build up quickly in localized areas, requiring fast-acting control mechanisms to prevent damage.

Given the complexity of modern chips, traditional thermal management techniques, such as using fan speed for cooling or globally reducing clock speeds, are no longer sufficient to ensure adequate system performance. Instead, advanced control strategies are needed to manage power and thermal characteristics on a much finer scale. These strategies often involve Dynamic Voltage and Frequency Scaling (DVFS), where the operating voltage and clock frequency of individual cores or regions of the chip are dynamically adjusted based on workload demands and thermal conditions. Similarly, clock gating can be used to selectively disable portions of the chip when they are not needed, further reducing power consumption and heat generation.

However, the need for fast and efficient control goes beyond simply adjusting voltages and frequencies. Modern HPC chips require real-time, multi-variable control strategies capable of reacting to changes in workload and thermal conditions within milliseconds [16]. These systems must not only respond to fast changes but also anticipate future evolutions. Advanced control techniques, such as Model Predictive Control (MPC) and Machine Learning (ML)-based algorithms, are being adopted with increased frequency to manage these dynamics (3.1).

In conclusion, the evolution of HPC chips toward many-core, heterogeneous architectures with chiplets and specialized accelerators has unlocked new levels of performance potential but also introduced significant thermal management challenges. As power densities rise and dark silicon becomes a growing concern, the need for advanced, fast-acting control mechanisms becomes paramount. The ability to manage thermal behavior and power consumption at a granular level is critical for ensuring both performance and reliability in modern systems.

1.4 Outline

This work investigates the modeling and control of power and thermal dynamics in HPC chips, with an emphasis on studying low-level control algorithms. The work is structured to gradually introduce background knowledge, model development, control challenges, control strategies, and their implementation, progressing from foundational concepts to advanced experimental setups. Each chapter builds on the previous one, presenting a structured approach that covers both theoretical and practical aspects of thermal and power regulation in HPC.

The outline of the thesis is as follows:

- **Chapter 2: Background and System Modelling**
  This chapter lays the groundwork for the thesis by exploring fundamental principles of thermal and power modeling in microelectronic systems. It outlines the structure of an HPC chip, focusing on key on-chip and off-chip components essential for the control. This foundational overview serves as the basis for constructing an HPC chip model, providing the basis to understand the control challenges and design decisions addressed in subsequent chapters.

- **Chapter 3: The Control Problem - Analysis and Requirements**
  This chapter begins by delineating the various layers within the HPC control hierarchy, then narrows down to define the specific control problem addressed in this work, defining key control variables and requirements. An analysis of challenges unique to HPC systems follows, particularly on system disturbances and delays, as well as the impact of actuator non-idealities and the exponential leakage power on control design. By describing in detail these requirements and challenges, the chapter highlights the necessity for specialized control solutions to effectively manage the complexity of modern many-core architectures.

- **Chapter 4: Control Algorithms**
  This chapter introduces various control algorithms tailored for dynamic power and thermal management. Starting with base techniques such as PID control and moving average methods, the chapter progresses to more advanced approaches, including fuzzy control, iterative root-finding methods, and Model Predictive Control (MPC). Each method is evaluated in terms of its suitability for specific control objectives, providing a comparative basis for understanding the strengths and limitations of different control choices in managing HPC requirements.

- **Chapter 5: Analysis and Comparison of Control Algorithms**
  In this chapter, the control algorithms introduced in Chapter 4 are evaluated and compared using thermal and power regulation metrics, as well as target compliance and performance indicators. The algorithms are tested under different workloads, thermal models, and architectural configurations, providing insights into each algorithm's effectiveness under different HPC operational conditions. Numerical results demonstrate that the proposed fuzzy-inspired iterative algorithm achieves up to a $5\times$ reduction on the maximum exceeded temperature compared to state-of-the-art methods, while providing an average 3.56% improvement in application execution runtime. Similar improvements are also observed over other base control techniques presented in this work. The fuzzy-inspired algorithm also delivers comparable application execution performance with the more computationally intensive MPC algorithms, while providing superior and more consistent power and thermal regulation results. The proposed linearized MPC implementation, although leading to a decrease in thermal and power regulation performance relative to a state-of-the-art configuration, achieves an average 15.56% improvement in target compliance metric. Finally, a test demonstrating the implementation of the gradient tracking algorithm for a set of distributed controllers under a shared constraint reveals a worst-case scenario delay of 30 iterations in achieving constraint satisfaction.

- **Chapter 6: Hardware in the Loop (HIL) Implementation**
  This chapter describes the implementation of a Hardware in the Loop (HIL)

platform developed to validate the proposed control algorithms under realistic conditions and provide a hardware-software co-design framework. Details are provided on the ControlPULP hardware, and the design process for the TPC firmware and the simulation integration. Experimental results validate the HIL capability to simulate realistic control scenarios for HPC chips, with insights into algorithm execution timing, communication testing, and overall control performance.

The resources and implementations developed in this work are publicly available in the following repositories:

- https://github.com/Ev3nt1ne/AechPeSi_lab: the Matlab simulation environment and control part;
- https://github.com/pulp-platform/control-pulp and https://github.com/pulp-platform/control_pulp_pcf: the proposed parallel LLC controller, including open-source hardware, firmware, and control algorithm;
- https://github.com/pulp-platform/pulp_hpc_cosim: the simulation environment written in C for FPGA-based co-simulation.

These repositories provide the necessary tools and resources for further exploration, replication, and extension of the methodologies presented in this thesis.

Chapter 2

Background and System Modelling

Modeling is a fundamental step in the journey of designing and developing complex systems. It allows investigating system behaviors under different conditions without physical prototypes or actual implementation, granting the flexibility of changing the design with ease and without additional costs. By developing mathematical representations of the system, it is possible to analyze its dynamic response to different inputs, disturbances, and changes in parameters, study stability properties, robustness, control performance, and many other criteria [63].

When developing models, it is important to identify the correct degree of accuracy and abstraction, as the level of detail and the appropriate assumptions have to be carefully tailored to the specific objective. While a highly detailed model may provide greater precision, it can also be computationally expensive and unnecessary for certain applications, leading to counterproductive results. Conversely, over-approximation or failures to identify relevant system non-idealities may lead to inaccurate predictions and potentially false results, undermining the validity of the control design.

There are various strategies to create a model, depending on the level of knowledge about the system and the available information. Black-box models rely purely on input-output experimental data and use identification techniques to generate models without any knowledge of the system's internal dynamics. White-box models, on the other hand, are derived directly from first-principle equations requiring a comprehensive understanding of the system's underlying physical laws. Gray-box models blend empirical data with some understanding of the system's physics, providing a middle-ground approach [151].

This chapter provides a detailed description of the HPC CPU and its computing chiplet structure from the perspective of the Dynamic Power and Thermal Management (DPTM) controller, aiming to develop an accurate white-box representation. The process starts with a comprehensive system description, which serves as the foundation for formulating the simulation model.

2.1 Related Works

In CPU modeling, the main interest has been in the thermal and power behavior of the system. Most studies [112, 108, 146], focus on modeling the thermal evolution, while Power modeling is frequently addressed as a complementary aspect of it. These surveys categorize thermal models in Finite-Element, Finite-Difference, and Spectral or Transform-Based approaches regarding white-box methods, while gray and black-box models are predominantly machine-learning-based or employ other identification techniques [26, 54].

Nearly all established and surveyed thermal modeling tools, such as HotSpot [141], ThermalScope [5], Sniper [117], Gem5 [31], IBM Turandot [102], BSIM [140], HSPICE [148], and other CAD softwares, are primarily designed to support hardware and architectural design. Tools like FloTHERM [138] and Therminator [166] focus on node-level cooling designs instead. Although these models are widely utilized in DPTM research, they present challenges when applied to control testing contexts, as extensive parameter adjustments are often required to make relevant thermal behavior emerge for the analysis of control algorithms. Additionally, many of these models lack integrated power dynamics or omit application-level execution factors, necessitating the use of complementary models, which increases both integration complexity and simulation time. Examples include McPAT [89] and Wattch [36], which emphasize core modeling, and tools like Orion [77] and CACTI [154], which focus on specific processor components. These limitations have led several studies to employ custom in-house models that simplify the thermal behavior for specific testing requirements.

In contrast, our modeling approach is orthogonal to these objectives, tailored specifically for "scaled" real-time simulations aimed at testing control algorithms. The proposed model abstracts beyond detailed hardware features, instead prioritizing time-domain evolution with a granularity greater than tens of microseconds. This abstraction allows for faster simulation speeds and captures key thermal trends essential for evaluating DPTM control algorithms.

2.2 The Physics of Microelectronics

The fundamental component at the base of HPC processors and other microelectronics is the transistor. By controlling the flow of electrical current, transistors enable the binary logic necessary for all modern digital computation [20]. This switching behavior is governed by an input voltage: below a certain threshold, the transistor remains off, blocking current flow, while above this threshold, it turns on, allowing current to pass through [91].

This switching capability is the key to digital circuits, enabling processors to execute complex calculations at high speeds. However, transistors are not perfect switches. Their operation is affected by several physical factors that impose limits on their performance and power consumption, including a temperature range of operation and considerations on energy losses [123].

Transistors experience two primary types of energy losses: resistive losses and leakage current. Resistive losses occur as current flows through the semiconductor material, where resistance causes energy dissipation in the form of heat. While these losses may be small, billions of transistors switching millions of times per second may accumulate significant heat generation. Leakage current, on the other hand, is always present even when the transistor is not switching. The effect occurs because the insulating layers are unable to fully block the electron flow, a problem that is further amplified in modern nanometer-scale designs. This unwanted leakage not only results in power loss but also adds to heat generation [103].

Heat generation poses a substantial challenge in transistor operation, as high temperatures can lead to significant degradation or failure of the component. With high temperatures, the materials within the transistor's channel can experience structural and electrical changes, impairing the ability to control current flow effectively. This thermal degradation not only increases leakage current but also disrupts the

precise switching behavior required for reliable operation. In severe cases, prolonged exposure to high temperatures can result in permanent damage to the component [46].

For HPC devices, which inherently generate considerable heat during operation, this thermal sensitivity needs to be taken into account. As transistors are pushed to operate at maximum performance, the excess heat produced must be carefully managed to avoid performance bottlenecks or system failure [132]. Advanced thermal management techniques and efficient cooling systems are essential to mitigate these effects and ensure that transistors operate within safe temperature ranges, preserving both the performance and longevity of the components.

2.2.1 Frequency-Voltage Relation

There is a physical relation between the frequency of a Processing Element (PE) (i.e. the speed of its transistors switching) and the voltage supplied to it for functioning [70]. In a Field Effect Transistor (FET) transistor-based digital circuit, toggling the transistor's state involves switching its gate voltage below or above a defined threshold. This process requires charging or discharging the transistor's gate capacitance. The rate of speed at which this change can happen affects the maximum frequency at which that digital circuit can operate. Higher voltages result in a faster slew rate when charging and discharging, allowing an increased switching speed [20]. Additionally, increasing the gate voltage beyond the threshold reduces the resistance of the FET's conducting channel, resulting in a lower RC time constant, which further accelerates the logic transitions [123].

Due to these physical characteristics, the maximum achievable frequency is dependent on the supplied voltage. Achieving higher frequencies necessitates increasing the voltage, though this relationship is sub-linear, meaning that frequency gains diminish as voltage is raised [70]. This direct dependency dictates how voltage and frequency can be changed dynamically. For instance, when increasing frequency, the voltage must be raised first to support the higher switching speeds, and the system must wait for the voltage transition to complete before the frequency can be safely increased. Conversely, when lowering the voltage to reduce power consumption, the frequency must be reduced first, as a lower voltage cannot sustain higher operational speeds without risking instability or errors. This sequential dependency adds complexity to the control mechanism, requiring synchronization between voltage and frequency transitions [83].

2.3 The On-chip Controller

Given the critical role of thermal management in maintaining transistor performance and reliability, modern high-performance processors rely on advanced dedicated on-chip Thermal and Power Controller (TPC) [49]. These controllers are designed to monitor and regulate temperature and power consumption dynamically, ensuring that the processor operates within safe limits even under high workloads.

State-of-the-art TPCs are implemented as embedded microcontrollers within the chip, counting one or more computing units and additional accelerators [110]. Due to their location, these controllers must be designed to operate with minimal power consumption and area overhead to avoid negatively impacting the overall performance of the processor. One of their primary responsibilities is Dynamic Power and Thermal Management (DPTM), which is achieved through Dynamic Voltage and Frequency Scaling (DVFS). By adjusting voltage and clock frequency of the chip,

the controller can optimize the power consumption and heat generation in real-time, ensuring that the system remains within safe operational limits [132]. To perform these functions, the TPC interacts with the physical components of the processor, continuously reading data from embedded sensors and controlling various actuators.

In addition to DPTM, these controllers collect data regarding the system state, aggregate and filter sensors data, and provide this information, upon request, to other agents, such as external management systems or software layers [78]. To enable this functionality, these controllers are equipped with a range of communication interfaces, allowing efficient interaction with both internal subsystems and external agents [110].

2.3.1 Local Controller for Distributed Control (LDC)

Recently, as many-core architectures have become more prevalent, the computational capability of the TPC has emerged as a bottleneck when managing large numbers of PEs [110]. To address this challenge, an emerging trend involves distributing the control workload across multiple -generally more limited- Local Controller for Distributed Control (LDC), each responsible for smaller clusters of PEs or even individual ones.

This distributed approach, made almost mandatory by the rising PEs count, introduces additional design considerations. On the hardware side, multiple LDCs require efficient inter-controller communication to ensure consistent and timely coordination. From a control perspective, developing robust distributed loops is fundamental to achieving global power and thermal objectives without sacrificing performance or reliability.

This distributed structure inherently enhances architectural modularity and scalability by allowing additional PEs and clusters to be integrated without significant redesign. Moreover, employing a hierarchical approach enables complex control algorithms to be implemented on a centralized higher-level advanced controller, while localized LDCs handle simpler, real-time decisions closer to the hardware. However, the effectiveness of distributed control heavily depends on communication delays. These delays can significantly impact the responsiveness of the system, particularly in enforcing power and thermal limits [38].

2.4 The Structure of a Processor

The Processor, or Central Processing Unit (CPU), is the heart of the computing power of an HPC system. It takes data and instruction from external memory and provides a result output. It contains several Processing Elements (PEs) (also commonly known as *cores*) and other modules to manage data and power lines. The CPU is installed on a motherboard, which is the board that hosts and connects all different components of a computing system, as well as the Power Delivery Network (PDN) [64]. The main components of the PDN are the Voltage Regulator Modules (VRMs) which take the power input from an external power supply and generate the correct voltage levels to give to each component on the motherboard [171]. Different components may have one or more VRMs, but it's not uncommon to find VRMs and power delivery lines shared across different components and parts.

Modern CPUs are no longer single-block silicon dies; instead, they consist of multiple interconnected chiplets [104]. Chiplet design emerged as a response to the

growing challenges in producing large, monolithic chips due to increased manufacturing costs, complexity, and diminishing performance gains at smaller process nodes [90]. As transistors became harder to scale and power consumption rose, the cost per area of silicon surged. By breaking a chip into smaller parts, manufacturers can improve yield, lower costs, and optimize each chiplet for specific functions (e.g. CPU cores, memory, or I/O), rather than using the same process node across the entire die.

In a chiplet-based structure, multiple chiplets are integrated within a single package, typically connected using silicon interposers or advanced organic substrates, enabling high bandwidth and reduced latency between components [85]. Some of these high-speed interconnects are AMD's Infinity Fabric [39] and Intel's EMIB [94].

In this work, the focus will be on computing chiplets housing General Purpose PEs, as these exhibit the highest power and thermal output, requiring more direct and complex control mechanisms. When modeling the thermal behavior of an individual chiplet, other chiplets are considered as external thermal sources. Depending on the design, chiplets may be separated by air or insulating material, resulting in near-thermal isolation [1], or they may be positioned in close proximity, establishing a stronger thermal coupling effect [16].

2.4.1 On-chip Components for DPTM

The primary components of the computing chiplets are the Processing Elements (PEs), each capable of executing independent tasks. Generally, a single chiplet contains multiple identical PEs. However, newer architectures particularly in consumer, mobile, and Internet of Things (IoT) markets, are adopting heterogeneous designs that integrate different types of PEs to improve specialization and energy efficiency [135]. While chiplets contain other elements such as caches (local fast memory), I/O controllers, and other subsystems such as the security subsystem [64], the primary focus of this work is on PEs.

The performance of PEs depends primarily on their Clock Frequency F, which dictates how quickly PEs' internal circuits switch states to complete operations. The two primary on-chip mechanisms that regulate clock frequency are the Phase-Locked Loops (PLLs) and the Dispatch Throttling (DT) [49]. PLLs are fundamental for generating the clock frequency for PEs, but they may be resource-intensive components, making it impractical to assign one to every PE in large systems, such as many-core CPUs. As a result, PLLs may be shared among several PEs [114]. However, modifying the clock frequency through a PLL incurs a delay during which the output frequency is unstable, potentially requiring to stall PE operations. A common solution to this problem is the use of dual PLLs: while one maintains the current clock signal, the other adjusts to the new frequency, ensuring smooth transitions. HPC chips, due to their performance requirements, may also contain a dual-PLL configuration for each PE [116].

In addition to shared PLLs, DT mechanisms are employed to minimize the need for frequent clock adjustments. These mechanisms are present in each PE and subdivide the clock signal at a more granular level, allowing finer control over the PE's execution speed without altering a shared PLL clock signal. This approach improves performance and energy efficiency across all configurations.

[1] This is due to the difference between fast silicon thermal time constant, and slower time constants of large air or insulating material gaps. More details are provided in section 2.6, while time constant computation is given from eq. (2.5).

Another set of on-chip components with which the TPC interacts are the sensors and the activity registers of the PEs. The most transversal is the Process, Voltage, Temperature (PVT) sensors [6], as the name suggests, measure three important metrics for the DPTM: the process variations, the supply voltage, and the temperature.

PVT Sensor

In semiconductor manufacturing, "process" refers to the series of steps used to create integrated circuits on a silicon wafer, including techniques like lithography, etching, doping, and metal layering to form transistors and other components [14]. Variations in the process can result in slight differences in the physical or electrical properties of the components, impacting performance and power consumption. Process variation can also be tracked to analyze the degradation of the component due to aging effects, and thus provide mitigation techniques [25, 43].

Although the supply voltage is regulated by off-chip components, such as the VRMs, which provide monitoring and control capabilities, it is important to measure the supply voltage also at the component level. Various factors, including on-chip mechanisms that dynamically modulate voltage, voltage droop, and fluctuations caused by activity spikes in other components sharing the PDN, can cause voltage level variations at the individual component level [126]. These deviations from the nominal value make it necessary to measure voltage directly at the PE to ensure a precise application of DVFS. Additionally, measuring both process variation and voltage allows for potential energy efficiency improvements by enabling fine-tuned voltage adjustments under certain conditions [55]. Further details on VRMs, on-chip voltage modulation mechanisms, and the relationship between voltage and frequency are explored in subsections 2.4.2, 2.4.3, and 2.2.1.

Temperature is the most important parameter measured by PVT sensors. By continuously monitoring it, these sensors provide real-time feedback to the TPC, enabling precise DPTM actions to maintain safe operating conditions. Accurate temperature measurements not only prevent thermal violations but also allow for the optimization of performance through dynamic adjustments to voltage and frequency. Furthermore, they contribute to long-term system reliability by mitigating risks associated with overheating and thermal stress, which can degrade hardware over time.

The number and placement of PVT sensors can vary significantly depending on design choices and system requirements. Generally, there is at least one sensor per PE, but there may be cases where a single sensor might monitor multiple PEs, or conversely, several sensors might be deployed per PE for more detailed monitoring of the hot spots. These design variations are driven by the need to balance area overhead, power consumption, and the level of monitoring granularity required for the specific application [99].

Despite their importance, PVT sensors provide generally noisy data. This noise arises from various factors, such as limitations due to design constraints, electromagnetic interference, and transient effects from switching activity in nearby circuits [6].

Activity Registers

In addition to sensor data, PEs store records of their operations in dedicated activity registers, providing insights into their execution performance. The data logged in these registers includes details about the types and frequency of operations executed, stalling or periods of inactivity, and transitions between active and sleep states [136].

This information is not only valuable for DPTM [45] but also for debugging and analyzing the performance of applications running on the PEs.

2.4.2 Off-chip Components for DPTM

HPC nodes contain several off-chip components. Most of them, such as Dynamic Random Access Memory (DRAM), parallel and graphics accelerators, and other peripherals, allow limited or no control by the TPC [64]. Consequently, these components fall outside the scope of this work, which focuses on the control action on the primary elements managed directly by the TPC.

One off-chip component important in this context is the Voltage Regulator Module (VRM). VRMs are responsible for converting the input voltage from the external power supply into appropriate voltage levels required by processors and other on-chip and off-chip components. Precision in voltage regulation is particularly important, as modern transistors are susceptible to even minor fluctuations [171]. To maintain safe operation, VRMs are equipped with protection features, such as safeguards against overvoltage, undervoltage, and overcurrent conditions, preventing potential damage to the components. Beyond ensuring stable voltage levels, VRMs also play an essential role in modulating voltage dynamically as part of DVFS control [64].

When shifting between discrete voltage levels, VRMs introduce delays or "transition times", necessitating either processor halting or enforcing a specific Frequency value for the transition duration, which can reduce application execution performance [116]. These transition times arise from the transient response limitations of the VRM, as its inductive and capacitive components add natural delays when adjusting the output to a new target. Consequently, voltage transitions involve waiting at a saturated frequency until the transition completes, making it reasonable to model it as a consistent maximum delay in the output response. This delay can impact overall system responsiveness, particularly under high dynamic workloads where frequent adjustments are necessary.

2.4.3 Shared Voltage Domains and Solutions

Similar to PLLs, VRMs are shared not only among PEs but also across chiplets and other off-chip components. This configuration establishes specific *Voltage Domains* within the system, where components must operate at the same voltage level, potentially leading to performance penalties or reduced energy efficiency [64, 12]. While PLLs use DT mechanisms for fast, decoupled signal level changes, recent voltage distribution architectures have incorporated additional components beyond VRMs, such as Fully Integrated Voltage Regulators (FIVRs) and Digital Low-Dropout Regulators (DLDOs), to avoid the highlighted issues [69].

FIVRs are on-chip regulators embedded directly within the die, allowing for precise, localized voltage control [41]. By regulating voltage close to the PEs, they minimize power losses from power distribution traces, simplify motherboard design, and offer rapid transient responses to sudden load changes. This configuration is especially beneficial for DVFS control, where per-core voltage adjustments are needed to optimize power and thermal efficiency.

DLDOs are linear regulators used to maintain stable, noise-free power at low voltage levels, often following the primary regulation from VRMs or FIVRs [164]. Unlike their analog counterpart, they rely on continuous feedback loops, allowing faster response and enabling their use for DVFS regulation [72]. DLDOs are less

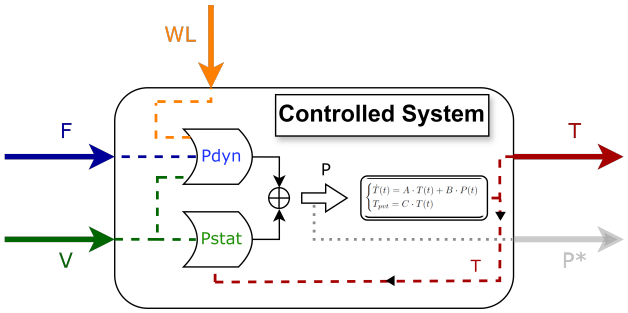


FIGURE 2.1: Representation of a PE from the control point of view. The power consumption $P*$ is grayed out because it is not directly measurable per single PE.

efficient with large input-to-output voltage differences, and dissipate excess power as heat.

Given their respective properties, FIVRs are commonly used to regulate voltage for larger components, such as chiplets; DLDOs, on the other hand, are more effective at a finer granularity, providing voltage adjustments for individual PEs or small groups of PEs.

2.5 Modelling the PEs

PEs are the primary contributors to power consumption and heat generation in the chip. Consequently, modeling their thermal and power characteristics is fundamental for testing and analyzing the TPC. In microelectronics, heat generation occurs due to the conversion of electrical energy into thermal energy during workload execution, mainly through the charging and discharging of internal capacitances within transistors [20]. Higher operating frequencies result in a higher rate of transistor switching, which consequently increases heat generation. The nature of the workload affects the number of logic gates activated to execute instructions, directly influencing the amount of heat produced. Voltage has a dual effect, contributing to heat generation through both active gate switching and passive leakage currents.

Logic-gate components are thus modeled as Multi-Input systems [16] as illustrated in Figure 2.1. The controllable input signals are the Frequency (F) and Voltage (V), while the Temperature (T) serves as the component's state. Another input to the system is the workload (ω), defined by the instructions being processed by the component. Differently from the control signals, the workload is not controllable by the TPC and acts as a form of disturbance or noise. The resulting model equation is:

$$\dot{T} = f(F, V, T, \omega, t) \tag{2.1}$$

where f is the non-linear mapping function of the model. All values vary with time, but the time dependency has been omitted for clarity.

Recently, HPC chip manufacturers have begun integrating instruction-gating modules that can be controlled by the TPC [9, 165]. These modules temporarily throttle certain types of instructions within specified time intervals, to filter out power spikes. However, despite the presence of these modules, the workload still acts as a source of noise. These modules only impose a limit on specific types of instructions, while the rest of the workload remains uncontrollable and unpredictable.

A conventional approach to divide and linearize system (2.1) is by introducing an intermediate term, the power consumption of the component (P) [146]. This variable is essential from a control perspective as it serves as one of the key objectives. The power consumption of a single component cannot be typically measured directly, but can still be estimated through the electrical energy conversion. However, the cumulative power consumption of multiple components can be monitored through off-chip VRMs [49].

By introducing power consumption, the model can be divided into two parts: a linear dynamic thermal model and a non-linear algebraic power model. In this structure, all inputs, including the workload, feed into the power model, while the output of the power model becomes the input for the thermal model. This separation introduces an approximation, as power consumption and temperature were originally coupled in the equation (2.1). In the new dual-model approach, this coupling can be preserved but is now treated as a relationship separated over time. This approximation holds if the simulation timing is much faster than the thermal evolution, making temperature changes negligible in the time delay [17].

The new modelling structure, illustrated inside the block of Figure 2.1, is particularly useful because thermal and power models exhibit different, and often opposite, characteristics. Power is typically modelled as an instantaneous, algebraic, non-linear mapping with multiple inputs. In contrast, the thermal model is a slower, dynamic, linear model, with multiple states and only one input [16, 146].

2.5.1 Thermal Model

Figure 2.2 illustrates the architecture considered for deriving the mathematical description of the thermal model of the system. It represents a chiplet integrated onto a silicon die over a carrier Printed Circuit Board (PCB), with a copper heat spreader placed over the active silicon devices to ease heat dissipation and provide a base for mounting the aluminum heat sink.

When modeling the thermal behavior, it is essential to consider the entire heat dissipation system. This includes the primary heat dissipation path, where heat is transferred through the heat spreader to the heat sink, and the secondary path, where heat is dissipated downwards through the substrate and into the PCB [142]. Additionally, it is important to account for not only vertical heat flow but also lateral heat distribution across the chiplet, which impacts the thermal dynamics within densely packed components [54]. Modeling in support of TPC algorithm design, the structure of the computing chiplet can be approximated as a grid of PEs and caches interconnected via communication lanes or Network on Chips (NoC) paths as illustrated in Figure 2.3.

The modeling process begins with the silicon and heat-spreader layers. The aluminum heat sink and the PCB are modeled with a different approximated approach provided in section 2.5.2 as an extension to the silicon and heat-spreader model. This is justified by three main factors: (i) the thermal time constants for the heat sink and PCB are three to four orders of magnitude slower than the faster thermal dynamics of the silicon [142], making them less significant from the controller's perspective[2] and for tests over a short duration of a few seconds; (ii) the thermal model of the heat sink is highly variable, depending on factors such as shape, material properties, and airflow conditions from the fans; and (iii) the lower surface of the cores can effectively be considered adiabatic [23, 146]. Consequently, modeling these components with

[2]See Chapter 3 for more details on why the TPC has to focus on the fastest dynamics.

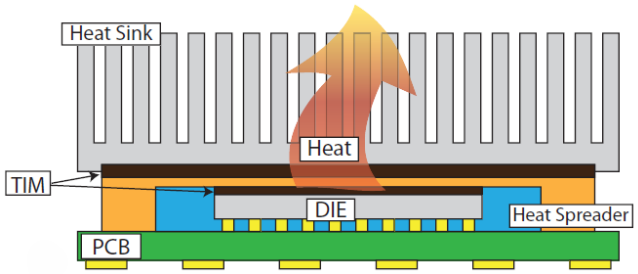


FIGURE 2.2: Vertical representation of the HPC Processor thermal structure. The main heat dissipation path is indicated by the red arrow passing through the Heat Spreader, the Heat sink, and two Thermal Interface Materials (TIMs) layers. The secondary heat dissipation path goes through the PCB below the die to the Motherboard.

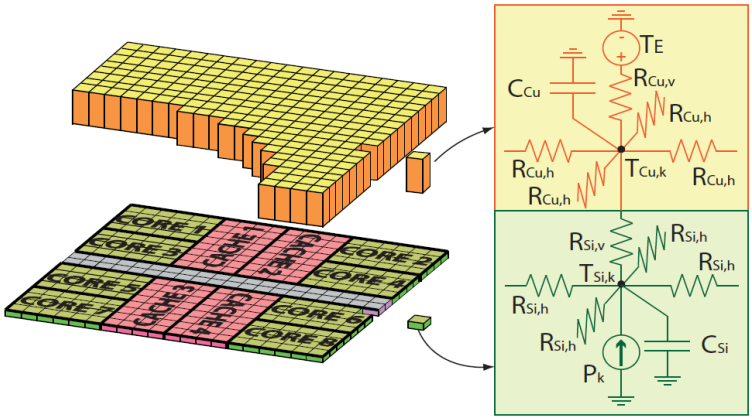


FIGURE 2.3: Example of finite elements spatial discretization of cores (PEs), caches, and interconnect. On the right, is the lumped parameters model of a single PE. The green part (below) concerns the silicon, with P_k being the consumed power generated by the PE, and the yellow part (above) concerns the heat-spreader, with T_E being the main dissipation path to the Heat Sink.

the same level of precision as the cores would introduce unnecessary complexity into the thermal model. For similar reasons, the interposer layer, required by the chiplet design is merged into the PCB model.

Starting from the physics first principles and applying Fourier's heat equation to the silicon and metallic layers, the following Partial Differential Equations (PDEs) are derived:

$$\rho_{Si} c_{Si} \frac{\partial T_{Si}(x,t)}{\partial t} = k_{Si} \nabla^2 T_{Si}(x,t) + q(x,t), \qquad \text{with: } x \in V_{Si}, t \in \mathbb{R}_{\geq 0} \tag{2.2}$$

$$\rho_{Cu} c_{Cu} \frac{\partial T_{Cu}(x,t)}{\partial t} = k_{Cu} \nabla^2 T_{Cu}(x,t), \qquad \text{with: } x \in V_{Cu}, t \in \mathbb{R}_{\geq 0}, \tag{2.3}$$

where $T_{Si}(\cdot)$ and $T_{Cu}(\cdot)$ are the temperatures of the silicon device and the heat-spreader defined in the open volumes V_{Si}, V_{Cu}, $q(\cdot) \geq 0$ is volumetric thermal power generated by internal sources (i.e. PEs' power output according to the model separation described in section 2.5), and ρ_{Si}, ρ_{Cu}, c_{Si}, c_{Cu}, k_{Si}, k_{Cu} are the density, specific heat and thermal conductivity of the two materials respectively.

To develop a model that is tractable for control design purposes, the PDEs in (2.2)-(2.3) can be "converted" into Ordinary Differential Equations (ODEs) by employing

Euler discretization of the derivatives along the spatial coordinate x. Clearly, this operation introduces some degree of approximation. However, resorting to the structural properties of the thermal system for which the *Maximum principle* [56] holds, it can be shown [155, 23, 141] that the feasibility of the thermal constraints can be guaranteed by discretizing the original PDEs in space and focusing on the hot spots of the die. For simplicity, n_c finite elements (cells) are chosen, each associated with a power source, i.e. a PE. Each finite element has two thermal states: one representing the local temperature of the silicon die and the other for the local metallic heat spreader. For the purpose of this model, the representation of the cache is simplified by treating it as a resistive and capacitive element while its power output is incorporated into the instruction-level power consumption.

Denoting with $\mathcal{N}\{i\}$ the set of all neighbors of the element i, then the differential equation associated with a generic element reads as:

$$\begin{aligned}\dot{T}_{Si,i} &= \frac{P_i}{C_{Si,i}} + \frac{T_{Cu,i} - T_{Si,i}}{C_{Si,i}R^v_{Si,i}} + \sum_{j\in\mathcal{N}\{i\}} \frac{T_{Si,j} - T_{Si,i}}{C_{Si,i}R^h_{Si,ij}} \\ \dot{T}_{Cu,i} &= \frac{T_{Si,i} - T_{Cu,i}}{C_{Cu,i}R^v_{Si,i}} + \frac{T_{Al} - T_{Cu,i}}{C_{Si,i}R^v_{Cu,i}} + \sum_{j\in\mathcal{N}\{i\}} \frac{T_{Cu,j} - T_{Cu,i}}{C_{Cu,i}R^h_{Cu,ij}},\end{aligned} \tag{2.4}$$

where $C_{Si,i}$, $C_{Cu,i}$ are the thermal capacitances, while $R^v_{Si,i}$, $R^v_{Cu,i}$, $R^h_{Si,ij}$, $R^h_{Cu,ij}$, $j \in \mathcal{N}\{i\}$ are respectively the vertical and horizontal thermal resistances. T_{Al} is the temperature of the aluminum heat sink, and P_i is the aggregated consumed power obtained from $P_i(t) = \int_{V_{s,i}} q(x,t)dV$. This representation, illustrated in Figure 2.3, draws a parallel to electrical circuit models, where heat flow is analogous to electrical current, and temperature difference corresponds to voltage. Thermal resistances and capacitances represent the opposition to heat flow and the ability to store heat, respectively, similar to their electrical counterparts.

These lumped parameters stem from the aforementioned spatial discretization procedure, with a detailed formulation provided in [113] and [141]. The corresponding equations are as follows:

$$\begin{aligned}C &= c_{th} \cdot \rho \cdot h \cdot w \cdot l \\ R^v &= \frac{1}{k_{th}} \cdot \frac{l}{h \cdot w} \\ R^h &= \frac{1}{k_{th}} \cdot \frac{h}{l \cdot w}\end{aligned} \tag{2.5}$$

where c_{th}, k_{th}, and ρ represent respectively the specific heat capacity ($[J/(kg \cdot K)]$), the thermal conductivity ($[W/(m \cdot K)]$), and the density of the material. h, w, and l denotes the height, width, and thickness of the component respectively.

Collecting all temperatures in a unique vector $T = (T_{Si,1}, T_{Cu,1}, \ldots, T_{Si,n_s}, T_{Cu,n_s}, T_{Al})^T$ the thermal model (2.4) can be compactly rewritten as:

$$\begin{cases}\dot{T}(t) = \mathbf{A}T(t) + \mathbf{B}P(t) \\ T_{Si} = \mathbf{C}T(t)\end{cases} \tag{2.6}$$

where $P = (P_1, \ldots, P_{n_c})^T$, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ follow directly.

2.5.2 Thermal Model Extensions

The aluminum heat sink and the PCB are modeled by introducing a single additional thermal state for each element. These states interact with neighboring layers through heat conduction, with the addition of a spreading resistance in series to account for the different contact areas between the layers [141]. The formula for the spreading resistances is derived in [87]. Albeit being an approximation [128], this approach aims to introduce slow time-varying temperature and dissipation shifts to study the impact of long-term dynamics and drifting states on the controllers.

To complete the two thermal dissipation paths, two additional states are added to the thermal model: one for the motherboard and one for the case's internal air. Additional resistances are added to emulate the presence of the two TIM layers [76]: the first between the cores and the heat-spreader, and the second between the heat-spreader and the heat sink. These TIMs enhance heat transfer across layers by eliminating air, which has a much higher thermal resistance. However, despite improving overall heat conduction, they still introduce a degree of thermal resistance between the surfaces.

Incorporating air as a thermal state requires two additional considerations: first, modeling heat convection between components instead of heat conduction, and second, accounting for the interaction with external air. Air heat convection occurs horizontally with the motherboard, and both horizontally and vertically with the heat sink.

For the heat sink, a multiplicative factor is introduced to account for its shape, which generally includes multiple fins that significantly increase the surface area for heat dissipation. This factor also accounts for active cooling provided by fans, which enhances convection by forcing air over the heat sink, accelerating heat removal [2]. Modern heat sinks are often more complex than a piece of metal, incorporating heat pipes that further improve thermal management by rapidly transferring heat away from the core components. Heat pipes operate by utilizing phase-change technology: a liquid inside the pipe evaporates as it absorbs heat, then condenses back into liquid when it reaches a cooler section of the pipe, effectively transporting heat with high efficiency [124]. However, for the purposes of this model, a simplified representation of the heat sink is sufficient. The goal is not to capture every detail of the thermal dynamics but rather to approximate the overall behavior and model a slow, long-term thermal drifts.

Modeling the interaction between the internal case air and the external environment is beyond the scope of this work. Accurate representation would require advanced fluid dynamics simulations to account for powerful, high-speed fans that push air either horizontally in rack-mounted systems or laterally in consumer cases, generating turbulent patterns. Instead, a simplified approach is adopted by introducing a basic heat-exchange factor to approximate the thermal interaction between the internal and external air. The external air temperature is also considered constant, assuming an infinite thermal capacitance. The vectors and matrices of the model (2.6) are extended with the inclusion of the described four states.

Fan speed presents an interesting challenge in thermal modeling, as it cannot be directly treated as an input signal but rather modifies the thermal model parameters, particularly the convective heat transfer rates. Modeling this interaction requires the use of time-varying or hybrid models, where the system remains linear, but the parameters that govern heat transfer are adjusted dynamically based on fan speed.

2.5.3 Power Model

According to the model separation described in section 2.5, one power source P_i is associated with each PE. This algebraic power model is a non-linear function of all the inputs to the HPC PE, as well as its current thermal state [20]:

$$P_i(t) = h_i(F_i(t), V_i(t), \omega_i(t), T_{Si,i}(t)), \tag{2.7}$$

where h_i is the nonlinear mapping function.

Various models can be employed for $h(\cdot)$ without altering the proposed structure of the system model. The power model used in this work is the most commonly used one [75, 146, 112, 103]. In this model, power is divided into two components: the dynamic power P_{dyn}, with the exclusive dependence on the Frequency F and the workload ω, and the leakage power (or static power) P_{stat}, which is independent of these factors and instead depends only on Voltage V and temperature T.

$$\begin{aligned} P &= P_{\text{stat}} + P_{\text{dyn}} \\ &= k_{s_0} + (I_{cc} V) \cdot \mathcal{K}(T_{Si}, V) + C_{eff} F V^2 \end{aligned} \tag{2.8}$$

where I_{cc} is the static current and C_{eff} is the effective capacitance which parametrizes the type of workload ($\omega(t)$) being executed on the PE. $\mathcal{K}(T_{Si}, V)$ is a non-linear function that encapsulates the dependency of the leakage power to the temperature and voltage of the component.

C_{eff} represents the portion of capacitance involved in transistor switching activity during computation. It accounts for the physical capacitances present in the transistors and interconnects, as well as the switching activity factor, which reflects how many transistors are toggling between states. Thus it depends on workload characteristics, circuit architecture, and the activity levels of different processor components [123].

In this work, an exponential relation based on [20, 130] is employed as the non-linear leakage component $\mathcal{K}$:

$$\mathcal{K}(T_{Si}, V) = e^{\left(k_v V(t) + k_T T(t) + k_{T_0}\right)} \tag{2.9}$$

where the k_* parameters are constant and computed on the critical values V_{MAX} and T_{MAX}.

Incorporating temperature dependency into the power model has several important implications. First, it preserves the original coupling (2.1) between temperature and power, though this relationship is now decoupled in time. Second, it introduces scenarios where, even with constant inputs (F, V) and workload ω, power consumption P may vary due to thermal drifts. Lastly, this addition models the thermal runaway scenario [158] where temperature and power increase each other in a positive feedback loop, leading to the irreversible damage of the component if those variables are not appropriately controlled by the TPC.

Dynamics in (2.6) combined with the algebraic power model in (2.8) describe the overall system thermal and power behavior.

2.5.4 Performance Model

In this work, the performance model is intended as two parts: modelling the instruction trace (or workload ω) and the mechanism to use this trace within simulations to compute C_{eff}, which is subsequently employed in the power model. Although from

the perspective of the TPC, the workload is treated as a noise signal [16], accurately modeling it is essential to achieving a dynamically variable testing setup. The rate of workload "consumption" depends on the PE's operational speed and changes in real-time according to the frequency control signal, making the performance model crucial for realistic testing analysis and comparison [20]. Furthermore, specific types of applications and workload scenarios are targeted for the controllers evaluation and comparison, and a well-developed performance model enables such characterization.

Modeling the workload with high precision is challenging, as it is architecture-dependent and often beyond the scope of this work. Workload dynamically changes with PE speed, which operates on a nanosecond scale, and is deeply influenced by architectural features such as out-of-order execution, deep pipelines, hardware multi-threading, and superscalar features [103, 12]. Additionally, execution characteristics like cache misses and synchronization barriers introduce non-deterministic memory latencies. Consequently, predicting the exact execution sequence of instructions in modern HPC architectures is nearly impossible without any degree of approximation.

Quantum-Based Workload Modelling

To effectively model the instruction trace in this work, a quantum-based approach is employed. Each "quantum" of instructions represents the smallest unit in the trace. These quanta exhibit distinct characteristics, variable sizes, and can be flexibly linked to each other. The primary assumption is that instruction density within each quantum is uniform, allowing temporal and causal dependencies within the quantum to be disregarded. This simplification is substantiated by the inherent unpredictability of exact trace execution and by the discrete time steps used in the thermal dynamic model. With a quantization matching the thermal model time step t_{th}, the introduced approximation is effectively masked by the time discretization required to run the simulation, reducing the need for finer granularity within the instruction trace. Additionally, PDN hardware smooths out any workload oscillations below $100\mu s$ [51], and for TPC control testing, the primary interest lies in application phases and average workload behavior.

The quantum-based approach enables a shift from modeling individual instructions to representing workload behavior with *instruction-characterized cycles*, inherently capturing characteristics such as Instructions per Cycle (IPC), memory stalls, cache misses, and SIMD/vector instructions. This cycle-focused method facilitates the use of real instruction traces from benchmarked executions and simplifies the creation of the simulation part of the performance model. To illustrate the utility of this shift, consider the case of the execution of a single instruction A which is characterized by an IPC of 0.5 and incurs five additional cycles due to cache misses. Within the quantum, this can be represented as two cycles attributed to A and five cycles to cache misses. In this way, the cycle representation preserves architectural complexity while allowing for an averaged approximation of execution behavior.

Before detailing the properties of each quantum unit, it's necessary to introduce a further characterization. Given the wide variety of instruction types and operand values—each potentially impacting power consumption uniquely—instructions are grouped into discrete power levels, each associated with its average C_{eff} value to determine the input for the power model.

Each quantum unit is defined by the following attributes:

- **N**: the number of cycles represented by the quantum
- **Cycles per Level (CPL)**: the distribution of cycles per power level

- **Wait Time**: a frequency-independent stall that generates the minimum C_{eff} level
- **Memory Boundness**: a factor describing how frequency impacts the consumption rate of the quantum
- **Additional Properties**: these may include synchronization barriers (for coordinating workloads across PEs), target changes (indicating the initiation of a specific command to the TPC), and other desired characteristics.

Simulation Mechanism for Quantum Consumption

In the simulation, a mechanism extracts from each quantum a number of cycles equal to the product of the PE's current frequency and t_{th}. The discretization time step of the thermal model t_{th} establishes the temporal resolution for the entire simulation. Depending on the frequency and the quantum size N, multiple quanta may be consumed within each t_{th} interval, or a single quantum may extend over several t_{th}. Consequently, the performance model must manage both the number of consumed cycles and the position within the quantum trace, as well as accurately translating wait time into cycle equivalents.

The memory boundness parameter determines how the operating frequency impacts cycle consumption within each quantum. This parameter effectively governs the relationship between frequency and cycle progression, directly influencing both energy efficiency and performance in control comparisons. The memory boundness value indicates the percentage of N cycles within the quantum that are unaffected by the operating frequency in terms of cycle consumption rate, and thus are independent of frequency changes. This serves as an alternative means to establish a pseudo-wait time property but with a defined CPL power level.

The execution time of these memory-bound cycles, and possibly the wait time, should instead be influenced by the uncore frequency [47]. The uncore frequency governs the operation of components outside the PEs, such as the cache, memory controller, and interconnects, which handle memory-bound operations. Adjusting uncore frequency can impact data access times and overall memory latency, thereby affecting the performance of memory-bound operations. However, this influence of uncore frequency is not yet implemented in the current model.

2.6 Model Implementation

In order to execute the simulation accurately, it is necessary to address the timing and dynamic requirements of each model. The performance model is relatively straightforward to implement, as its quantum-based design (2.5.4) allows it to adapt to various resolutions, albeit with approximations depending on the chosen discretization. The power model is defined by an algebraic equation (2.5.3) and thus it does not impose specific dynamics on the simulation. However, timing constraints are introduced by the hardware elements within the PDN, requiring power changes in the order of $100\mu s$ to be managed by the TPC controller [51].

The thermal model, being dynamic, imposes stricter timing requirements. Processor thermal time constants vary across different components, with values in the order a $\sim 10s$, $\sim 0.1s$, $\sim 1ms$ [26], relative to the heat sink, heat-spreader, and silicon dynamic respectively. To avoid the complexity of solving continuous models through ODEs, the thermal model is discretized. This discretization also aligns well with the

digital nature of the TPC controller, facilitating the execution of both within a unified environment.

Considering these factors, a time discretization interval (and a simulation timestep) t_{th} between $10\mu s$ and $50\mu s$ is chosen. This timestep: (i) captures power fluctuations relevant to the controller, (ii) avoids overly coarse approximation of workload dynamics ($10\mu s$ at $5GHz$ represents 50.000 cycles), and (iii) effectively simulates thermal dynamics by running at a rate approximately two orders of magnitude faster than the shortest thermal time constant in the system, while striking a balance with computational requirements.

Proper attention must be given to accurately modeling the system's non-idealities introduced by real sensors, actuators, and system architecture. A more detailed discussion is provided in section 3.3. Of particular importance are the delays affecting the TPC execution, including the sensors' zero-order hold delay, communication delays, and actuator delays, all of which influence and degrade the control response [58]. Moreover, the shared structure of actuators for VRMs and PLLs, and the relationship between frequency and voltage F-V described in section 2.2.1 impose additional constraints. If these factors are not appropriately considered, they may lead to inaccurate control performance results.

The parameters for the thermal and power models used in the simulation are derived from our study of the EPI Rhea1 processor, supplemented with data from other works [130, 23, 21]. This combination of empirical data and validated literature ensures that the models reflect realistic HPC chip behavior, providing a robust foundation for testing and evaluating control algorithms.

2.7 Conclusion

This chapter has presented a comprehensive framework for modeling computing chiplets, focusing on its application in simulating and evaluating the TPC and its corresponding control algorithm. The modeling methodology has been examined in detail, outlining the rationale behind approximations and the emphasis on accurately representing elements that are critical for control interactions. The modeling approach accounts for system dynamics, control algorithm execution periodicity, and physical constraints, ensuring relevance in dynamic power and thermal management scenarios.

The discussion has detailed the key components of an HPC chiplet, integrating physical descriptions with functional modeling. The computing chiplet is decomposed into three distinct submodels—performance, power, and thermal— with their own characteristics. The methodologies used to derive these models, including the governing physical principles and necessary simplifications, have been systematically presented.

The thermal model is structured as a spatially discretized representation of the heat transfer dynamics within the silicon die, heat spreader, and associated cooling mechanisms. By leveraging lumped-element modeling, the formulation captures both vertical and lateral heat dissipation paths while maintaining computational tractability for real-time simulations. The power model incorporates both dynamic and leakage power components, highlighting the exponential dependency of leakage power on temperature and voltage, which introduces additional complexity in control tasks. The performance model is constructed based on an instruction-quantum abstraction, allowing flexible, modular, and computationally light workload characterization, introducing approximations that don't impact TPC testing.

Beyond the formulation of these models, this chapter has emphasized the practical implementation aspects, including time discretization, the choice of simulation parameters, and considerations regarding sensor noise, actuator response times, and other real-world non-idealities. The selected time step balances accuracy and computational efficiency. The modeling framework introduced in this chapter serves as a foundation for the subsequent analysis of control strategies. The next chapter builds on this framework by defining the control problem, formulating objectives for the TPC, and analyzing the impact of non-idealities. The insights gained from this simulation provide the necessary groundwork for designing and evaluating advanced control algorithms tailored for HPC chiplets.

Chapter 3

The Control Problem - Analysis and Requirements

The development of control strategies requires a precise understanding of the system's scope, requirements, and inherent limitations. Achieving an effective control design depends on accommodating the unique characteristics of the system, including non-idealities, time delays, disturbances, and constraints on the control signal. Therefore, it is essential to clearly define the control problem, encompassing its operational scope and the limitations imposed by factors such as computational resources, available information, and environmental conditions.

Control systems can be divided into different layers or hierarchies, each characterized by varying degrees of abstraction, time scales, and system knowledge. Lower layers often deal with direct interactions with physical components, requiring rapid responses to changes in the system's state. Higher layers, in contrast, may focus on strategic optimization, incorporating broader system-wide information to make decisions over longer time horizons [143].

This strategy is known as cascade control design, where multiple controllers are arranged in a nested configuration, allowing finer and more precise control of complex systems. In this arrangement, the output of one controller, known as the outer controller, serves as the reference or setpoint for another controller, called the inner controller. The inner controller directly influences the system's actuators, allowing faster response to disturbances and more effective control over specific system variables. This nested approach provides enhanced stability and reduces the effects of disturbances that might affect the system between the two control loops. It also improves system performance by separating and controlling dynamic behaviors at different time scales on different hierarchy levels.

In HPC architectures, control is implemented through a cascade of control blocks, each operating within distinct time domains, addressing different requirements, and according to diverse abstractions. These blocks can be broadly categorized into three main types as represented in Figure 3.1: the Low-Level Controller (LLC), the High-Level Controller (HLC), and the Global-Level Controller (GLC) [24, 12].

As the name suggests, the LLC operates directly on the hardware, and it is the only type of controller with unmediated access to the physical components and sensors of the system [122]. Positioned at the lowest level of the control hierarchy, the LLC has limited computational power and access only local system information. Designed to meet strict dependability and safety requirements, LLCs perform real-time control tasks with response times ranging from microseconds to milliseconds, ensuring fast reactions to changes in operating conditions and workloads. Consequently, LLCs typically rely on simple and reactive control strategies [49].

HLC instead operates at a higher level of abstraction within the software stack, leveraging greater computational resources but with considerations on execution

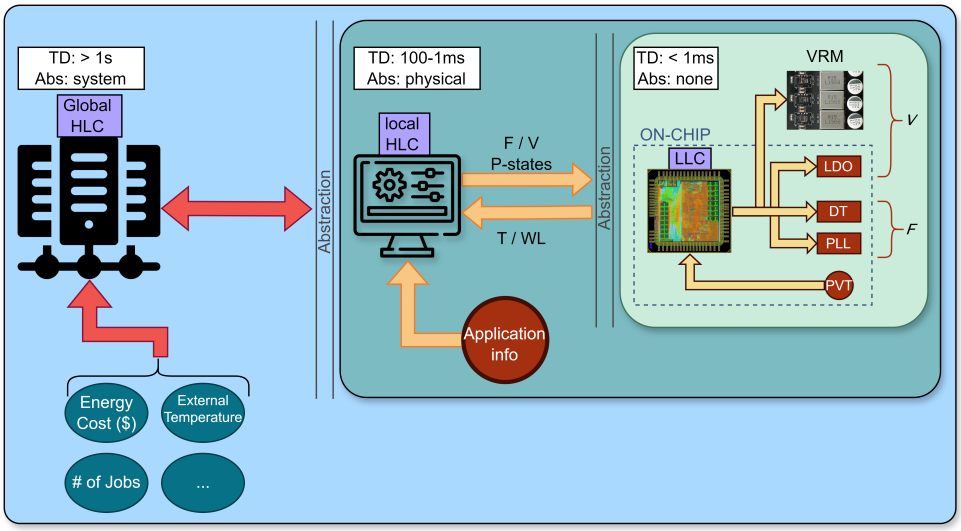


FIGURE 3.1: **Cascade Control Architecture in an HPC System.** The three control blocks operate in separate time domains, each with its own system abstraction and external data sources. The LLC (inner light green) manages low-level actuators (VRM, DLDO, DT, PLL) based on target operating points from the local HLC and retrieves sensor data (PVT and others) for feedback. The HLC (middle dark green) processes system and application data to compute operating points. The GLC (external light blue) integrates system-wide and environmental data to define performance and energy efficiency targets for each local HLC.

overhead [16]. For this reason they have slower time scales, typically in the range of milliseconds to seconds. Due to their execution context, HLCs have a broader knowledge of the system, including workload characteristics, application performance metrics, and the state of software execution. Their main objective is to optimize the trade-off between performance, energy efficiency, and user experience. HLCs generally employ more complex and computationally intensive algorithms that take into account factors such as workload distribution, system state, and inter-system coupling. They often rely on predictive control algorithms, such as MPC, to anticipate system needs and optimize performance [103].

At the highest level of the hierarchy, the GLC abstracts even further from the system hardware, focusing on managing and coordinating resources across the node or even the entire computing center [95]. More similar to the HLC but with a more managerial role, the GLC operates at a time scale of seconds with an even broader scope, overseeing global power and thermal budgets, ensuring system stability, and managing job dispatching and resource allocation. Their focus is on long-term optimization and coordination rather than fine-grained control [163].

Other types of control agents, such as the Board Management Controller (BMC), may also be present in an HPC system. The BMC functions as a hybrid between LLC and HLC, being a controller on the motherboard and interacting with hardware components [59]. However, from the LLC's perspective, these agents merely set input targets and constraints, making them functionally similar to HLCs in the context of control strategies [16].

As outlined in the Introduction 1 and in section 2.3, this work focuses on the LLC. In accordance with the separation property of cascade control, it is important to understand the specific requirements of each controller to effectively differentiate

their roles and scope. Therefore, in this chapter the LLC control problem is described, distinguishing it from the characteristics and state-of-the-art approaches of other types of controllers. By understanding the requirements and control challenges, it becomes possible to compare the effectiveness of various control strategies and propose more effective solutions.

3.1 Related Works

The majority of state-of-the-art research on HPC control focuses on HLC algorithms. Much of the literature emphasizes prediction and optimization over a deep understanding of the physical system's nuances, architecture, and constraints. When research is tested on real systems, unless it pioneers work on newer RISC-V chips, it automatically leans on an underlying LLC. Furthermore, several of these algorithms address only a single metric, such as power or temperature, leaving other problematic conditions to be managed by the LLC layer [122].

While HLC studies are valuable, comparable research specifically for LLC control is limited. Some earlier studies could be loosely classified as LLC research [132], but these typically involved basic control techniques targeting single-core systems with significantly lower leakage power and fewer nonlinear behaviors.

Modern HLC controllers [49, 103] are devised to be implemented within the OS routines or as user-level applications that run alongside the main system workload. Leveraging application-class PEs, HLCs generally have access to greater computational resources than LLCs, albeit limited by the contention with the executing workload, but their access to system internals is restricted for security reasons [122]. Current HLC designs support advanced control algorithms aimed at *prediction* and *optimality* [145, 43]. These algorithms consider a broad characterization of the system, including current executed workload and external conditions [49], rather than focusing solely on the physical and architectural aspects of the system. Consequently, HLCs tend to prioritize energy efficiency and optimal application performance tradeoffs over strict power and thermal constraints. To interface with the physical system, HLCs frequently utilize abstractions provided by protocols like Advanced Configuration and Power Interface (ACPI) [136].

Model Predictive Control (MPC) is commonly applied in HLC design for thermal management, using optimized trajectories calculated from sophisticated cost functions [23, 155, 96, 160, 161]. A group of HLC works [35, 27, 45] focuses primarily on enhancing performance and energy efficiency through application analysis, as well as optimized thread scheduling and placement. Model identification techniques [22, 43] including ML-based methods [97] support the development of complex and accurate HLC controllers that adapt in real-time. A substantial body of research exists on ML-based HLC controllers [115] that optimize not only energy efficiency and performance but also task scheduling and thread management. Although there is a growing interest in lightweight HLC controllers, this trend primarily applies to mobile platforms [30, 48], which differ significantly from HPC systems in terms of PE count and power scale.

Research on GLC is also flourishing [95, 57, 3], with a comprehensive focus on efficient resource allocation, global thermal management, and energy efficiency. Overseeing and coordinating multiple HLC and LLC controllers across numerous nodes, GLCs optimize overall data-center performance, energy consumption, and cooling efficiency. These controllers utilize large-scale optimization and predictive

models to balance workload distribution, while real-time monitoring and adaptive control strategies help mitigate thermal hotspots and reduce operational costs.

3.2 The LLC Control Problem

Due to the increasing specialization of processors, the advent of dark silicon designs, and many-core configurations, it is no longer feasible for all cores to operate at their maximum performance simultaneously [88]. Doing so would result in excessive power consumption, surpassing the VRMs' limits and the capacity of the PDN [64]. This would also lead to thermal constraints violation, as the heat generated by the transistors would exceed the capacity of the cooling system, with horizontal thermal coupling between neighboring PEs further exacerbating the issue.

In accordance with the separation principle of the defined cascade structure [143], the control problem of the LLC can be formulated as follows::

- Apply as close as possible the inputs provided by the outer control loops
- Manage the faster dynamics, including the PEs' fastest thermal response and power spikes
- Ensure that all safety requirements are met, including transistor thermal limits, VRMs power and current thresholds, and PDN constraints

Additionally, the secondary objective is to:

- Optimize the performance-to-energy efficiency trade-off based on received targets.

Supplementary constraint requirements, such as a target PE temperature or a lower power budget, may be requested from the LLC for optimization decisions at higher levels of the control. However, selecting the most stringent requirement (i.e., the lowest value) is sufficient to fulfill all requirements. Thus, to simplify the notation moving forward, this minimal constraint assumption will be adopted when discussing power budgets and thermal limits. In doing so, it is important to note that the constraints become time-varying.

The control problem for the LLC can then be formulated as follows:

$$\begin{aligned} &\min_{F_a} \left| F_T(t) - F_a(t) \right|_{\mathcal{R}}^2 \\ &\text{subject to : } \sum_{i=1}^{n_c} P_i(t) \leq P_{\mathrm{B}}(t) \\ &\qquad \sum_{i=1}^{n_{\mathcal{D}_j}} P_i(t) \leq P_{\mathcal{D}_j}(t), \quad i \in \mathcal{D}_j, \quad j = 1, \ldots, n_d \\ &\qquad T_i(t) \leq T_{\mathrm{L}}(t), \quad i = 1, \ldots, n_c \end{aligned} \tag{3.1}$$

where $F_a = (F_{a1}, \ldots, F_{an_c})^T$ are the applied Frequencies to the n_c PEs, $F_T = (F_{T1}, \ldots, F_{Tn_c})^T$ denotes the target frequencies provided as operating points from the higher layers of the control system, and $\mathcal{R} \in \mathbb{R}^{n_c \times n_c}$ is a symmetric positive definite weight matrix with $|\cdot|_R$ its corresponding norm. P_{B} represents the total power budget of the node, while $P_{\mathcal{D}_j}$ are the power budgets for the VRM domains. n_d refers to the number of $\mathcal{D}_j$ voltage domains such that $\sum_{j=1}^{n_d} n_j = n_c$. Lastly, T_i is the temperature of the i^{th} PE and T_{L} is the thermal limit.

In this control framework, the variables P_B and $P_{\mathcal{D}_j}$ exclude off-chip components such as DRAMs, storage devices, and Peripheral Component Interconnect (PCI) components such as Graphic Processing Unit (GPU). However, subtracting from those variables the power consumed by off-chip components, over which the TPC has limited control, can effectively account for their power consumption without adding complexity, as these values are already considered time-varying based on earlier considerations. Additionally, this work primarily focuses on controlling the PEs within a computing chiplet, which is why the control of off-chip components is not considered.

3.3 Control Challenges

Numerous state-of-the-art solutions have been proposed to address the control problem (3.1) [132, 49]. However, many of these solutions do not fully consider all the control challenges involved. Primarily, the described system operates as a multi-unit, non-linear Multiple-Input Multiple-Output (MIMO) system, where the control objectives are interconnected. Then, the control signals are both coupled and discretized at specific values. Moreover, the system is subject to high-frequency, unpredictable noise, arising from the unknown workload $\omega(t)$ [16].

Additionally, the control algorithm must function at a frequency in the range of $2 - 10kHz$ on an embedded microcontroller. This frequency is dictated by the constraints imposed by the PDN and the fastest thermal time constants (as outlined in section 2.6 and in section 3.3.4). Maintaining this operational frequency ensures effective regulation of temperature and power capping, in accordance with commonly applied control principles [58].

3.3.1 Exponential Leakage Power Analysis

Leakage power P_{stat} is a critical factor in modern HPC systems, and its modeling can significantly impact power management strategies. As introduced in section 2.5.3, this work presents an exponential leakage power model (EXP-PS), which offers a distinct contrast to the linear models (LIN-PS) used in other studies [146, 75]. Figure 3.2 illustrates the comparison between the EXP-PS model (2.9) represented by the blue plane, and the LIN-PS model shown with the magenta plane.

In the first plot, leakage power values are plotted against Voltage (V) and Temperature (T). At high voltage and temperature, typical in HPC systems, the EXP-PS model predicts values up to 10 times greater than those of the LIN-PS model. The top-down view of the leakage power graph in the bottom left of Figure 3.2 illustrates this further, revealing that for about half of the operating points, the EXP-PS model produces results similar to or lower than the LIN-PS model. However, in high-power and high-performance scenarios, the two models diverge significantly.

In the remaining three plots, the graphs compare leakage power to dynamic power P_{dyn}, with the yellow plane representing the lowest workload (LOW-PD) and the red plane the highest (HI-PD). It can be observed that in low-power states—where the EXP-PS and LIN-PS models overlap—the leakage power is comparable to LOW-PD. However, in high-performance states, leakage power increases to levels that approach those seen at moderate dynamic power workloads, eventually reaching values similar to HI-PD.

As a result, the contribution of exponential leakage power is negligible in low-power states. However, as the system moves into high-power states, leakage power

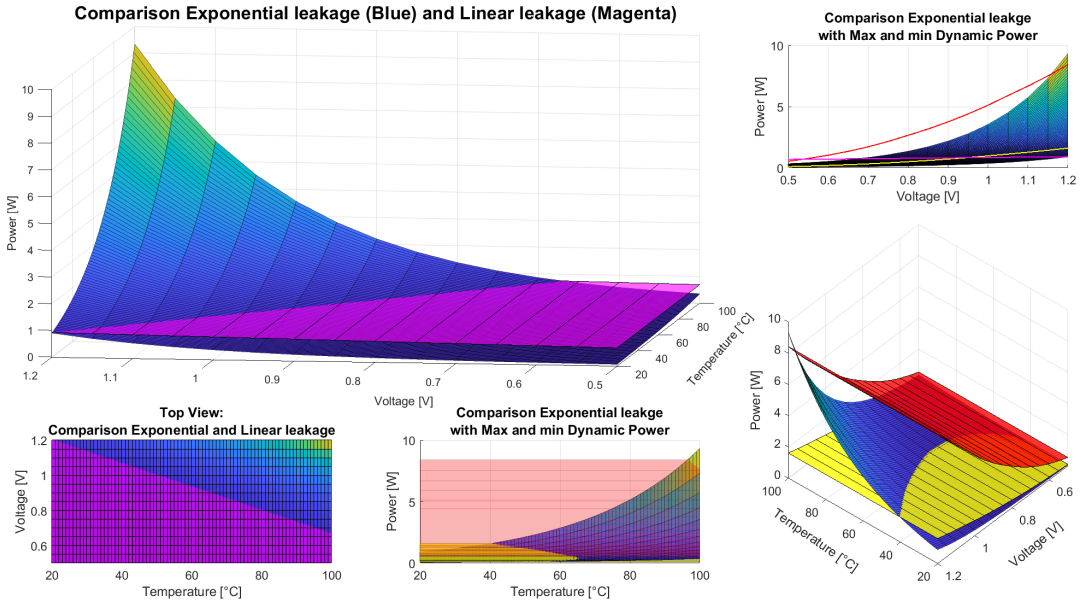


FIGURE 3.2: Multi-view comparison between the proposed model of the leakage power characterized by an exponential relationship with Voltage and Temperature (blue surface), the same model of leakage power with no exponential relation (magenta surface), and the minimum and maximum dynamic power (yellow and red surface respectively).

grows substantially, accounting for nearly half of the total power consumption. This behavior is driven by the non-linear relationship between leakage power, voltage, and temperature, posing a challenge for simple control algorithms.

The EXP-PS model not only complicates the design of the control, but also strengthens the coupling between the two key control objectives: satisfying power and temperature constraints. This interconnection results in a positive feedback loop, where a rise in temperatures leads to higher leakage power, further increasing the temperature. This feedback loop is the primary cause of the thermal runaway scenario described in section 2.5.3, posing a significant control challenge in HPC systems.

3.3.2 Control Signals

As discussed in section 2.2.1, there is a relationship between the two control signals F and V, where certain frequency levels can only be sustained with a corresponding minimum voltage. However, these signals exhibit different characteristics: frequency is directly linked to the execution performance of the application running on the Application-class Processors (APs), while voltage serves as the primary control signal for reducing power consumption [1] to meet the given constraints. Since the signals are coupled, simply increasing the frequency while lowering the voltage is not feasible. Instead, it is crucial to find an optimal trade-off operating point [12, 70]. Indicating with $\mathcal{D}_j$ the j^{th} voltage domain that supplies the component c_i, and omitting the time dependency for ease of reading, The F-V relation can be described as:

$$F_i \in [\, F^S_{\min},\ F_{\max}(V_{\mathcal{D}_j}, T_i)\,] \qquad \text{with } c_i \in \mathcal{D}_j \tag{3.2}$$

[1] According to (2.8) and (2.9), Voltage (V) affects both dynamic and static power components with a superlinear dependency.

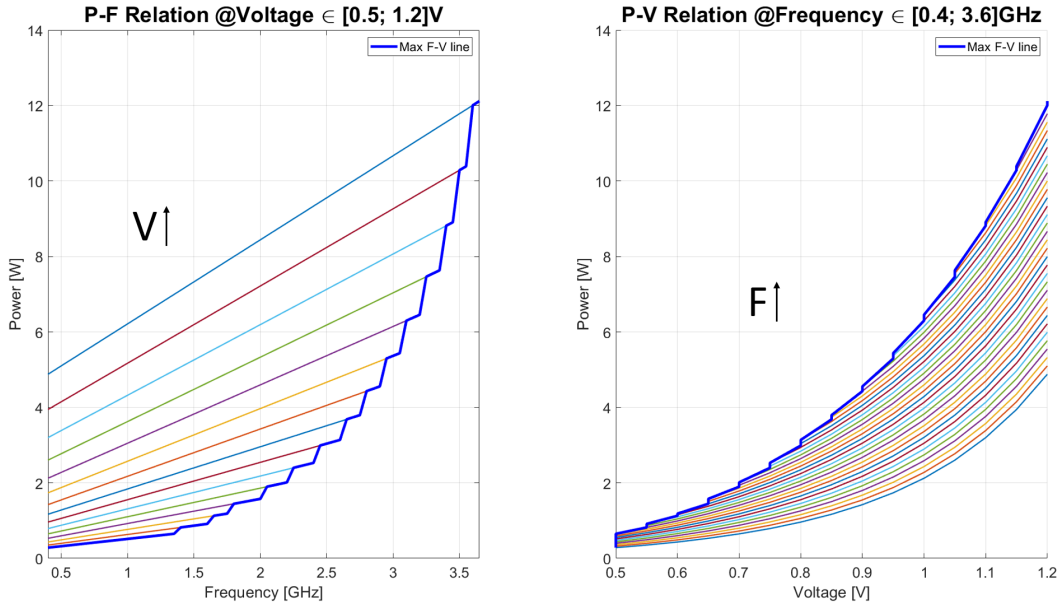


FIGURE 3.3: The PE power consumption $P(t)$, as modeled by (2.8), as a function of frequency F (left) and voltage V (right) at 75°C. Each graph displays only the operating points that satisfy the feasibility conditions defined in (3.2). Different lines correspond to different fixed values of the variable not explicitly plotted in the respective graph.

where $F^S_{\min}$ is the minimum PE frequency of the system.

Further complexity arises from shared actuators, response times, transition delays, and synchronization, as outlined in subsections 2.4.1, 2.2.1, and 2.4.3. Frequency adjustments can be made more rapidly due to the fast response time of the DT, but these adjustments will have a lower impact on power consumption compared to voltage changes. This is because the voltage impacts powrer consumption quadratically and exponentially (2.8). These challenges will be analyzed in the following subsections.

Moving forward, it is useful to define the target power P_T, which controllers generally use as the control variable. P_T is computed from the operating point pair (F, V) and an estimate of the workload ω, using the power model. Since P_T is linear with respect to the system to be controlled (as detailed in section 2.5), linear controllers may be employed. Additionally, P_T is directly related to both thermal and power requirements providing further simplification. From P_T, the Target Efficient Power (TEP) can be defined, representing the target power consumption of a PE at the most efficient frequency and voltage pair [42], based on the Frequency-Voltage relationship (3.2). This should not be confused with the related concept of Thermal Design Power (TDP), which represents the average power that can be dissipated as heat over a defined time window, ensuring that the system can sustain steady-state operation without exceeding thermal limits. [64].

Power steps

The relationship between power, frequency, and voltage plays a crucial role in the performance and efficiency of HPC systems. Understanding how power changes with frequency and voltage adjustments is key to optimizing power consumption. Figure 3.3 illustrates the progression of the TEP for a PE based on the frequency-voltage relationship (3.2) in an HPC system. In the left figure, the target power P_T is plotted against the frequency, while in the right one, it is plotted against voltage.

The bold blue line represents the TEP, and additional lines show P_T values at lower performance-efficiency levels. When reading graphs, it is important to remember that the actuator outputs are quantized on specific values or *levels* [12].

In the left part of Figure 3.3, each straight line represents P_T values at a fixed voltage level, since power is linearly affected by the frequency F. As voltage levels increase, the slope of each line becomes steeper, with each line ending at the maximum rated frequency for its respective voltage level. Observing the blue TEP line, which maximizes frequency for each voltage level, reveals that the power difference (the *power step*) is much steeper when changing voltage levels compared to adjusting frequency within a fixed voltage level. For example, the power jump along the TEP line is more than $3\times$ greater at $2.2GHz$ and $5\times$ greater at $2.7GHz$ than the slope of P_T. This indicates that while frequency adjustments within a voltage level have a smaller impact on power and thermal output, changing voltage levels results in significant power and thermal deviations.

Using the data from Figure 3.3 along the TEP blue line, increasing the frequency by $50MHz$ from $2.85GHz$ to $2.90GHz$ at $0.9V$ raises P_T by $62.7mW$. However, the same frequency increase from $2.90GHz$ to $2.95GHz$ (which requires a voltage level increase as $2.9GHz$ is the maximum allowed frequency for $0.9V$) raises P_T by $740.7mW$-over $10\times$ more. It should also be noted that these values are measured at fixed temperatures and do not account for additional power increases caused by temperature rising from the higher heat generation. The control algorithm must account for these differences when selecting the operating point of each PE.

Voltage Domains

The complexity of the previous example increases when considering shared actuators, particularly the PLL and VRM. The primary issue with the PLL is the limited number of available frequency combinations, as for each individual PE only the DT can be controlled directly, and not the PLL. Additionally, global changes to the PLL clock may cause slight frequency shifts even with a constant control signal. However, as described in section 3.3.2, small changes in frequency do not cause significant power fluctuations.

The shared VRM introduces a more significant challenge, as seen in the right plot of Figure 3.3, where P_T is plotted against the voltage V. The blue line represents the same TEP as in the left figure, while the curved lines represent different fixed frequency values. The shape of the lines is a result of the quadratic and exponential dependence of P_T on V in the chosen model (2.8). When voltage is shared among PEs, it creates an additional coupling between the control signals. Increasing the voltage raises the power consumption (and temperature) of all PEs in the domain. Conversely, reducing the voltage caps the maximum frequency, potentially limiting the performance of PEs that were not previously constrained by power or thermal limits. The control algorithm must account for this coupling when distributing power or setting PE frequencies.

For instance, consider PE A, which is capped at $3.8W$ due to thermal limits and operates at the TEP point $\{2.75GHz, 0.85V\}$. If the domain's voltage is increased to $0.95V$, perhaps due to another PE's frequency request, PE A would need to reduce its frequency to $1.85GHz$ to maintain the same target power, resulting in a performance loss.

Consider the opposite scenario where PE A has thermal headroom and could operate at its maximum frequency. If PE B, sharing the same voltage domain, needs

to reduce its voltage to dissipate accumulated heat[2], PE A would be constrained to the same maximum frequency as PE B, leading to underutilization of the available power budget. Due to VRM sharing, a controller adjusting the voltage must consider whether the resulting ΔP_T could cause power budget violations or temperature overshoots in any PEs within the domain. Otherwise, the thermal control system would need to reduce the frequency (and performance) to maintain safe operating conditions.

To address this challenge and reduce the complexity of control algorithms, it is important to introduce fine-grained voltage reduction mechanisms similar to how the DT operates for PLLs. Some of the solutions currently used are presented in section 2.4.3.

3.3.3 Disturbances and Oscillations

In HPC systems, various disturbances can significantly affect control accuracy and performance. There are three major types of disturbances in a typical PE control scenario: instruction variability, process variation and parameter drift, and sensor noise. The common control strategies in the literature can address the magnitude of sensor noise discussed in section 2.4.1. However, it is important to note that such disturbances hinder precise power and workload identification at the individual PE level when relying solely on PVT sensors, particularly in higher frequency domains.

Parameter and process deviations in transistor technology are largely due to variations in the manufacturing process. Previous studies report that these deviations can cause up to a 10% variation in the current flowing through the manufactured devices [25, 168, 14]. Additionally, parameters can change gradually over time due to aging effects. These variations influence the power consumption and thermal response of the PE, meaning that control algorithms must account for these differences and avoid controlling all PEs uniformly.

Instruction variability, instead, directly impacts the dynamic power component of power consumption. Although power spikes are smoothed by the PDN [51], this variability still results in a constantly changing noise input to the controlled system. This disturbance is unpredictable, varies at a frequency faster than the control algorithm's response time, and contributes significantly to overall power consumption [110], as illustrated in Figure 3.2. Consequently, the system may experience power consumption fluctuations of several watts per iteration, even when the input values of $F(t_k)$ and $V(t_k)$ (along with the estimated target power P_T) remain constant. For this reason, the control strategy should be characterized by a careful balance between reactivity and filtering to effectively manage this variability.

Power and Thermal

Power steps caused by voltage level quantization and non-linearities, described in section 3.3.2, can cause significant oscillations around the thermal and power constraint setpoint. These oscillations occur because the high amplitude of power steps prevents precise tracking of the setpoints, leading to oscillating control signals and throttling unless a margin around the setpoint—within which slight deviations are acceptable—is allowed. This issue is more prevalent when dealing with a large

[2]From the left-hand side of Figure 3.3, a PE running at the minimum frequency ($0.4GHz$) and at the maximum voltage, generates a considerable power consumption ($5.52W$). This is similar to the power generated by the same PE at $\{3.0GHz, 0.95V\}$. This means that, even when running at minimum frequency, a PE may need to reduce its voltage level to reduce its temperature.

number of PEs within a single shared actuator domain, where changes to the VRM or PLL values can increase the oscillations, which are further intensified by the thermal coupling of neighboring PEs.

Oscillations are also caused by disturbances, particularly instruction variability, which can cause fluctuations in power consumption and subsequently affect temperature. This issue is exacerbated by controller delays described in section 3.3.4. High-amplitude oscillations may lead to violations of thermal and power capping constraints, reduce performance over time if not properly filtered by the control algorithm, and place additional stress on the hardware, accelerating its aging process [46].

3.3.4 Controller Delays

In digital control systems, such as the LLC and other embedded microcontrollers, control actions are executed at discrete intervals, in contrast to continuous-time systems. This discrete-time nature introduces inherent delays into the control loop, as the controller updates its output only at specific time steps rather than continuously. These delays arise from the periodic execution of the control algorithm, where computations are performed in batches between sampling instants [58]. As a result, any change in the system's state is not immediately acted upon but instead processed at the next scheduled update, creating a lag between the system's response and the controller's output.

Additionally, to ensure consistent output generation, digital discrete controllers apply the computed output at the beginning of the next periodic execution cycle [58], resulting in an additional delay of $1t_s$, where t_s represents the controller's periodic execution interval.

These delays can negatively affect control performance, particularly in fast-changing or highly dynamic systems, where timing mismatches can lead to instability, oscillations, or sluggish response. The longer the delay, the more challenging it becomes to maintain precision in controlling the system, as its state evolves faster than the controller's ability to respond [58].

In the control framework considered in this work, it is key to analyze how power consumption is particularly affected by delays. As described in eq. (2.8), power is directly related to the workload ω; however, as discussed in section 3.3.3, the workload exhibits high variability and acts as a high-amplitude noise. When measured, the workload is only observed during the subsequent control interval, which introduces a delay. If there is a sudden shift in the workload's average value—a common occurrence [45]—the controller's response will inevitably lag. Considering the additional delay for the consistent output generation, the total delay becomes $(1 + v)t_s$, where $v \in (0, 1)$ represents the point within the control interval at which the workload variation occurs. This delay contributes to power and thermal oscillations, with one of the worst scenarios being an oscillating workload in phase with the controller interval.

To reduce thermal oscillations, the controller interval t_s should be less than half of the fastest thermal time constant. This rule, commonly applied in control systems [58], also ensures that the control algorithm can manage the maximum delay in power consumption variations, preventing excessive PE temperature fluctuations.

Mitigating power oscillations caused by sensing delays typically requires the use of hardware power management features built into the PE by the manufacturer. Historically, Intel addressed this issue by forcefully reducing the frequency when specific high-power workloads were detected, as reported in [136]. More recent designs incorporate microarchitectural mechanisms, such as dynamic micro-op scheduling,

to delay the dispatch of high-power instructions, effectively distributing the associated power increase over multiple control intervals [9, 165]. These mechanisms can effectively mitigate power oscillations, but only within short time windows, as extending them further would significantly compromise overall performance.

Actuators Delays

As described in sections 2.4.1 and 2.4.2 and in [116], the actuation of control signals is not instantaneous and can introduce additional delays into the control system. These delays are significant when PLLs and VRMs levels are adjusted, due to their required settling times. To avoid introducing additional oscillations or performance degradation, it is essential for controllers to either predict an optimal operating point—thus minimizing the need for frequent adjustments—or to apply low-pass smoothing filters to the control signals.

Loop Separation

Due to the differing characteristics of the two main control variables—temperature and power—control algorithms can be divided into two different loops, each with its own timing. Temperature evolves dynamically according to specific thermal time constants, with the fastest evolution typically in the order of $1ms$ [141]. In contrast, power consumption changes are instantaneous and unpredictable, requiring faster control adjustments.

To address this, a fast control loop operating at approximately $10KHz$ can be implemented to regulate power fluctuations and manage hardware instruction dispatch mechanisms. This high-frequency control is essential for mitigating power spikes and ensuring compliance with the PDN [51]. In contrast, a slower loop running at around $2KHz$ is sufficient for thermal management, as temperature variations occur over longer timescales. By dividing the control algorithm into these two loops, the system can effectively balance the distinct demands of power and thermal management [15].

3.4 Conclusion

This chapter has extended the modeling framework presented in Chapter 2 by integrating a detailed analysis of the real-world challenges associated with controlling an HPC system. The discussion has focused on key system non-idealities, including shared actuator constraints, sensor noise, system delays, and power steps, all of which significantly influence the effectiveness of control algorithms. These factors introduce complexities that must be carefully considered when designing controllers capable of achieving robust power and thermal management.

A fundamental aspect is the distinction between the Low-Level Controller (LLC), High-Level Controller (HLC), and Global-Level Controller (GLC), clarifying their respective roles in the control hierarchy. By reasoning within the separation principle, it is established how each layer operates at different time scales, with distinct responsibilities and objectives. A key issue in prior research is the lack of a clear distinction among these layers, often leading to confusion in control objectives and improperly designed strategies that mix real-time enforcement with high-level optimization policies. By precisely defining these roles, this work focuses exclusively on LLC analysis, ensuring that control strategies are evaluated in their correct execution domain, without conflating responsibilities that belong to the HLC or GLC. This clear

separation provides a robust basis for defining the control objectives and thus enables a precise comparison among algorithms.

Among the system non-idealities, this chapter highlights the importance of understanding the exponential nature of leakage power. The comparison between a linear power model (LIN-PS) and an exponential leakage power model (EXP-PS) revealed significant implications for control design. In low-power states, the two models yield similar predictions, but as power and temperature increase, the exponential model diverges substantially from the linear approximation. This discrepancy poses a major challenge for linear controllers, which may fail to anticipate the rapid escalation of power and temperature in high-performance states. The phenomenon is further exacerbated by the thermal and power coupling, where temperature-induced leakage power growth leads to uncontrolled heating, necessitating advanced regulation mechanisms that can preemptively mitigate such effects without overcompensating in lower power states.

Additionally, the chapter has addressed the issue of power step quantization due to discrete voltage and frequency levels. The transition between voltage levels introduces abrupt power shifts, leading to oscillatory behavior if not carefully managed. These oscillations can compromise stability, degrade system performance, and reduce overall efficiency. The introduction of actuators, sensors, and controller delays further exacerbates these effects, as control actions are not immediately applied, requiring predictive strategies to anticipate system behavior.

Another critical aspect is the impact of voltage domain coupling, where multiple processing elements share voltage regulation. This constraint imposes limitations on Dynamic Voltage Scaling (DVS), creating trade-offs between performance and regulation. The analysis demonstrated that controlling frequency and voltage independently is not always feasible, as changes in voltage affect all processing elements within a domain, leading to performance penalties and potential inefficiencies in power distribution.

By framing the control problem in this realistic context, this chapter establishes a foundation for the next section, where various control algorithms will be introduced and analyzed. The insights gained from this discussion will inform the reasoning behind control technique choices.

Chapter 4

Control Algorithms

Designing effective control algorithms for HPC systems requires careful consideration of the system's characteristics, specific control requirements, and inherent constraints. As modern HPC architectures continue to evolve toward many-core and heterogeneous designs, the need for advanced, adaptive control strategies becomes increasingly essential [80]. These considerations influence design decisions, such as whether a centralized or distributed control approach is appropriate, and which system sensors and actuators are to be used.

Safety and reliability criteria are major concerns in this context, and controllers should maintain temperatures and power consumption within limits, while optimizing their control action such as minimizing performance degradation due to thermal throttling or power capping. To meet these requirements, a certain degree of real-time responsiveness and robust steady-state performance must be achieved. Satisfying these demands might require the integration of predictive and adaptive mechanisms that can preemptively adjust to fluctuating system states, as purely reactive approaches may prove insufficient [29].

Understanding the characteristics of the underlying system is thus fundamental for an effective control design. HPC systems exhibit unique behaviors, such as nonlinear power-temperature relationships, time delays introduced by actuators, and the impact of workload variations on energy consumption and thermal distribution (3.3). Accounting for these characteristics ensures that control algorithms are robust and effective. For example, actuator delays might necessitate the implementation of predictive control models, while the inherent nonlinearities of power and temperature dynamics could be addressed through advanced techniques suitable for non-linear systems. Additionally, the architecture's physical structure, such as the distribution of sensors and actuators, must be considered during the design phase.

This chapter explores various control algorithms, from conventional methods to more advanced strategies, and discusses how each approach is tailored to the unique demands of HPC systems.

4.1 Related Works

To the best of the author's knowledge, four major players in the HPC industry currently employ proprietary LLC systems: Intel, AMD, IBM, and ARM.

Intel's approach centers on monitoring power consumption at the functional block level, with dynamic budget allocation across various components [52]. Power is estimated using a model that incorporates leakage information, voltage, temperature, and workload estimates [157]. Intel's control is implemented through an Exponential Moving Average (EMA) algorithm over multiple time windows in the order of seconds [133]. According to [66], additional thermal control features operate in

parallel with power capping. One of Intel's flagship management tools is the Running Average Power Limit (RAPL), a hardware feature enabling fine-grained monitoring of energy consumption and power/thermal capping across multiple power domains, covering CPUs, GPUs, and DRAM [78]. Monitoring is conducted via model-specific registers (MSRs), counters integrated within the architecture [68].

AMD, on the other hand, employs a distributed Proportional Integral Derivative (PID) control structure [39, 32]. In AMD's system, thermal and power control are handled independently, using frequency as the controlled variable, while a voting-box mechanism resolves inter-component coupling by choosing the lowest frequency. Given the design similarities, AMD's power and thermal management approach resembles Intel's RAPL from a structural perspective.

IBM's Power9 OpenPOWER firmware is available as open-source [73]. The control uses a voting-box mechanism to select the most limiting frequency across components, similar to AMD's approach. Temperature and power are controlled independently, with frequency as the controlled variable. Unlike AMD, however, IBM's control firmware follows a centralized design, collecting data to a central unit and performing a single algorithmic computation with moving average filters and PID-like controllers. One component in IBM's voting-box mechanism incorporates workload estimation to enhance control precision.

ARM has also released its System Control Processor (SCP) firmware as open-source [8]. ARM's control design is a cascaded structure featuring weight-based power distribution, with a thermal dispatching layer preceding the power reduction layer. In this setup, power serves as the controlled variable, with a PID controller enforcing thermal capping and a two-stage power distribution system with the second layer distributing unused power to the cores via an accumulation variable. ARM uses two independent TPCs based on the Arm Cortex-M7 microcontroller: the SCP for power management and the Manageability Control Processor (MCP) for communication functionalities [122]. In ARM-based System-on-Chip (SoC) architectures, interactions with the operating system are managed through the System Control and Management Interface (SCMI) protocol [92]. SCMI provides a set of OS-agnostic, standardized interfaces for managing power domains, voltage, clock, and sensors via a shared, interrupt-driven mailbox system.

In addition to traditional LLC approaches, advanced control strategies such as Fuzzy logic and MPC have also been explored for DPTM in HPC environments. Despite being typically employed as HLCs, these techniques have shown significant potential in optimizing thermal and power performance. Notably, fuzzy control strategies have demonstrated robustness and adaptability in varying thermal conditions [101, 48], while MPC has been praised for its predictive capabilities and ability to handle complex system dynamics and constraints [23, 155]. However, most MPC implementations have focused exclusively on thermal capping, often neglecting power constraints and key system interactions such as domain coupling and the frequency-voltage relationship. Nonetheless, these approaches pave the way for more sophisticated, integrated control solutions, bridging the research gap between HLC and LLC controllers.

4.2 Baseline Algorithm: PID and Moving Average Techniques

In the development of control strategies, the initial algorithm implemented was structured as a two-stage process [15]. This design, called Baseline Algorithm (BA), incorporated both a heuristic-based power distribution mechanism and a PID controller,

reflecting the most commonly used structures and algorithms in state-of-the-art methodologies. The distinguishing feature of this approach was the power distribution algorithm, which adapted dynamically to the temperature of PEs, coupled with a cascade arrangement of the two control layers, rather than a parallel structure commonly used in conventional designs. Consistent with the separation framework outlined in section 2.5, the initial stage (PW-DIST) was responsible for power capping and allocation among the PEs, followed by the PID algorithm (TH-PID) in the second stage, tasked with regulating temperature.

In BA, the chosen control variable is the target power P_{T}. As outlined in section 3.3, this variable is linear with respect to the system being controlled and can incorporate all constraint requirements related to both temperature regulation and power capping. The control action consists of four main steps:

1. **Initial Power Estimation (*Conv2P*)**
   This initial conversion step uses F_T and ω to estimate the power consumption of each PE, denoted as P_{est}, through the controller's power model h_{est}.

$$P_{\mathrm{est}} = h_{\mathrm{est}}\big(F_T,\, \omega,\, \cdots\big) \tag{4.1}$$

2. **Power Capping (PW-DIST)**
   The second stage applies power capping based on the total estimated power consumption $\sum_i^{n_c} P_{\mathrm{est},i}$. The control system ensures that the total power allocated across the PEs remains within the predefined limits, distributing the available power budget P_{B}. The reduction is activated when $\Delta P_{\mathrm{C}} > 0$, with:

$$\Delta P_{\mathrm{C}} = \sum_{i=1}^{n_c} P_{\mathrm{est},i} - P_{\mathrm{B}} \tag{4.2}$$

   Power is distributed across PEs based on the thermal room, defined as the difference between the thermal limit T_{L} and temperature T_i of each PE. In other words, frequency reductions are not uniform but are instead directly proportional to the PEs' temperatures, preventing power allocation to PEs that would be thermally capped by (TH-PID). Furthermore, assuming that the HLC layers provide accurate Dynamic Frequency Scaling (DFS) F_T targets, power should not be overly reduced for PEs running high-performance workloads, even if their temperatures are elevated. The α-Weighted Heuristic Algorithm (αWHA) follows three steps:

   (a) Compute the thermal-based α weights for each PE:

$$\alpha_i = \frac{1}{T_{\mathrm{L,i}} - T_i} \tag{4.3}$$

   (b) Normalize the α weights:

$$\bar{\alpha}_i = \frac{\alpha_i}{\sum_{i=1}^{n_c} \alpha_i} \tag{4.4}$$

   (c) Reduce PEs's power as:

$$P_{\mathrm{T},i} = P_{\mathrm{est},i} - \bar{\alpha}_i \Delta P_{\mathrm{C}} \tag{4.5}$$

   This ensures that the total power consumption of all PEs $\sum_i^{n_c} P_{\mathrm{T},i}$ does not exceed the target power limit P_{B}.

3. **Thermal Capping (TH-PID)** In the third step, thermal regulation is applied using a PID controller for each PE. This implementation employs only the proportional and integral parts of the PID, enhanced with error banding techniques to reduce oscillations. Since the PID in this structure provides a corrective action on top of a control signal, rather than tracking a setpoint, the output of the PID is added to the original P_T value. To prevent increasing power adjustments, the output of the PID is subjected to upper saturation at the value of 0.

   When the controller's output is saturated, the error may continue accumulating in the integrator despite the system no longer responding to it, causing the integral term to grow excessively. This accumulation can lead to overshoot or slow recovery. To mitigate this, an anti-windup mechanism is included to saturate the accumulator of the integral component [13]. The implemented PID structure is shown in Figure 4.1.

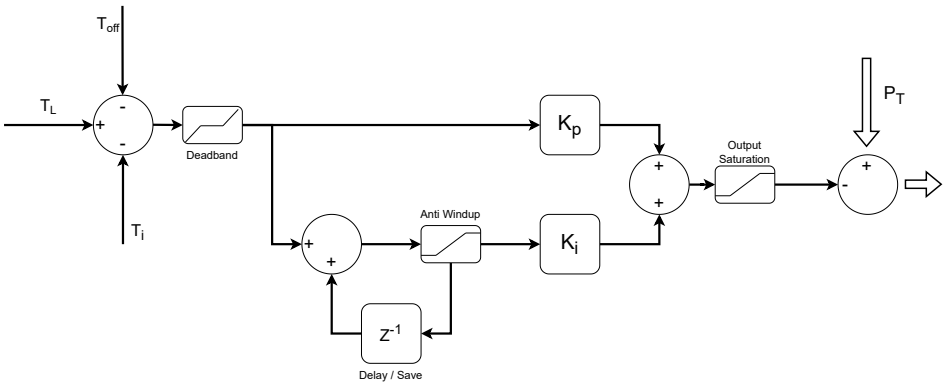


FIGURE 4.1: Structure of the PID used in BA, including the Proportional and Integral (discrete) components, along with Deadband, Anti-windup, and Output Saturation features. The output is subtracted from the P_T output of PW-DIST.

   All PID parameters have to be tuned according to the specific characteristics of the HPC system being controlled. The tuning process should find a balance between achieving a fast response to power fluctuations caused by workload variations, minimizing the impact of the measurement noise from PVT sensors, and avoiding large overshoots or oscillations [10]. Overshoots and oscillations may be particularly problematic as they could result in constraint violations and performance degradation respectively.

4. **Frequency and Voltage Selection (*Conv2F*)** In this final conversion step, the target power P_T of each PE is mapped back to an operating point defined by the (F, V) pair. Several strategies can be employed to achieve this mapping, with the most common approach being the use of a look-up table (LUT).

$$\{F, V\} = \eta_{est}(P_T) \tag{4.6}$$

   where $\eta_{est}()$ is the inverse of the $h_{est}()$ mapping function in the initial conversion Conv2P.

Notably, in the entire cascade structure, each step of the algorithm is applied independently to each PE. The only instances where PEs are considered collectively occur during the PW-DIST stage, where the estimated power contributions (P_{est}) are summed, and during the normalization of the α weights.

This control structure is similar to the basic ARM control scheme described in section 4.1, with the notable difference being the absence of a 'power recovery' feature that would reallocate power capped during thermal regulation. However, this limitation posed minimal impact, as power is distributed preferentially to PEs further away from their thermal limits.

The BA was initially tested with a simplified simulation, where the voltage was fixed, a LIN-PS power model was employed, voltage coupling in shared voltage domains and secondary thermal effects such as thermal drift were not considered. Under these simplifications, the algorithm demonstrated promising results, effectively managing power distribution and thermal regulation. The findings are presented in [15] and [17].

4.2.1 Enhanced Baseline Algorithm

While the BA demonstrated promising results under simplified conditions, several enhancements were required to adapt the algorithm to a more realistic HPC model. In particular, introducing variable voltage and accounting for the voltage-frequency relationship, as discussed in section 2.2.1, along with the management of multiple voltage domains, necessitated significant modifications to the initial design. In particular, a DVS method needed to be incorporated.

The final Conv2F step of BA involves converting P_{T} into the corresponding (F, V) pair. This is a multi-variable problem with a non-unique solution. In cases without voltage coupling, the pair could be selected using a LUT to get the (F, V) pairs along the TEP line. However, this approach generated oscillations in the control outputs, which degraded the overall controller performance. The origin of these oscillations is analyzed in section 3.3.3.

In shared voltage domains, this approach may result in an overly conservative policy. As emphasized earlier, the algorithm treats each PE individually. When voltage coupling constraints are introduced, a single voltage level must be selected for the entire domain. If one of the higher levels is chosen to preserve the performance of the faster PEs, this can lead to thermal constraint violations or cause severe performance penalties to thermally constrained PEs, as detailed in section 3.3.2. Conversely, selecting the lowest voltage respects thermal constraints but sacrifices the performance of faster PEs due to the maximum achievable frequency for a given voltage, while also leaving some of the previously allocated power unused.

These considerations highlight that enforcing voltage coupling constraints *a posteriori*, i.e. after the final step, does not yield the best results. An alternative approach for determining the (F, V) pair is to fix one of the two variables through all the steps of the algorithm, allowing the controller to properly consider thermal and power reduction on the other variable, and the Conv2F step to produce a unique result.

Voltage was selected as the fixed variable, as it tends to change less frequently, while frequency has a wider range of available values within a fixed voltage level, according to the frequency-voltage relationship (3.2). Additionally, voltage is the variable subjected to coupling constraints and the one that produces the largest power steps, making it the most likely to cause oscillations, which this approach seeks to minimize.

The first control step, Conv2P, already requires a fixed voltage value to compute P_{est}, meaning that the domain voltage must be selected as the first step of the control algorithm. Thus, instead of determining the domain voltage based on the required target frequency F_T, an Exponential Moving Average (EMA) low-pass filter on the past PE applied frequencies F_a is employed for selecting $V_{\mathcal{D}_j}$. This ensures that

the voltage selection is based on a more realistic frequency (F_{V-MA}), minimizing excessive performance cuts for both high-performance and thermally constrained PEs. Additionally, this introduces a degree of optimization, as the η^{-1} mapping function loses one degree of freedom (the fixed V). By employing this approach, the maximum voltage across the PEs in the domain can be selected, while the low-pass filter helps smooth out rapid voltage changes, reducing oscillations.

After each iteration t_k of the control algorithm, for each PE, the difference between its target frequency $F_T(t_k)$ and its applied frequency $F_a(t_k)$ is added to an accumulation value $F_{\text{MA}}(t_k)$ specific to each PE, through a forgetting factor λ_{MA}. Over several iterations, the value $F_{V-MA} = F_T - F_{MA}$ converges to track the average frequency selected by the control algorithm over a period ($\mathcal{T}_{\text{MA}}$) dependent on the choice of λ_{MA}.

In the Conv2P step, $F_{\text{MA}}(t_k)$ is used to select the voltage level for the domain, providing a more realistic estimation of the power consumption of each PE. However, the target frequency $F_T(t_k)$ is still used in the computation of the estimated power P_{est}, allowing the control algorithm to increase the applied frequency when thermal and power headrooms are available. At each iteration, the Enhanced Baseline Algorithm (EBA) executes for each domain $\mathcal{D}_j$ the following operations in the improved Conv2P step:

$$F_{V-\text{MA},i}(t_k) = F_{T,i}(t_k) - F_{\text{MA},i}(t_{k-1}) \qquad \forall\, i \in \mathcal{D}_j \tag{4.7}$$

$$V_i(t_k) = f_V\big(F_{V-\text{MA},i}(t_k)\big) \tag{4.8}$$

$$V_{\mathcal{D}_j}(t_k) = g_{\mathcal{D}}\big(\, V_i(t_k)\,\big) \tag{4.9}$$

$$P_{\text{est},i} = h_{\text{est}}\big(F_{T,i}(t_k),\, V_{\mathcal{D}_j}(t_k),\, \cdots\big) \tag{4.10}$$

where $f_V()$ is a step-wise monotonically increasing function that computes the minimum voltage required for a given frequency F, ensuring that the condition in (3.2) is satisfied. Essentially, for each F, f_V provides the corresponding V closest to the TEP line. The function $g_{\mathcal{D}_j}()$ determines the domain voltage based on the $n_{\mathcal{D}_j}$ voltages across the domain. The final Conv2F is changed as:

$$F_{a,i} = \eta^{-1}\big(P_{\text{T},i},\, V_{\mathcal{D}_j}(t_k),\, \cdots\big) \qquad \forall\, i \in \mathcal{D}_j \tag{4.11}$$

$$F_{\text{MA},i}(t_k) = (1-\lambda_{\text{MA}})F_{\text{MA},i}(t_{k-1}) \;+\; \lambda_{\text{MA}}(F_{T,i}(t_k) - F_{a,i}(t_k)) \tag{4.12}$$

Equation (4.12) represents a discretized form of a low-pass filter, where the forgetting factor λ_{MA} is defined as the ratio between the controller sampling time t_s and filter time constant τ. The value of τ is selected to reject the system state perturbation caused by workload variation and intercept the main trends. In this work, τ is set to $\tau = 1.25 \cdot 10^{-2}$.

The EMA approach can reduce performance during transients following a steep change in one of the set points or system conditions, depending on the chosen forgetting factor. To mitigate this effect, a phase detection strategy could be used to reset $F_{\text{MA}}(t_k)$ when a phase shift is detected or a set point adjustment is required by HLCs, significantly decreasing the "adaptation time".

A second modification to the BA addresses the significant difference between the estimated power $\sum P_{\text{est}}$ and the power consumption measured from the VRMs P_{Mes}. This deviation arises from the use of a simplified power model in the algorithm h_{est} compared to a more complex model in the simulated system. This difference can cause the BA to deviate from the power budget set point by a large and variable margin. The proposed solution employs an EMA adaptation filter where the difference between

the total $\sum P_{\text{est}}$ and the measured power P_{Mes} is included into PW-DIST, to account for the power model mismatch while minimizing high-frequency noise. The modified PW-DIST is:

$$\begin{aligned} \Delta P_{\text{Tot}}(t_k) &= P_{\text{Mes}}(t_{k-1}) - \sum_i P_{\text{est},i}(t_{k-d-1}) \\ \alpha_{pw}(t_k) &= \begin{cases} \alpha_{\text{up}} & \text{if } \Delta P_{\text{Tot}}(t_k) \leq 0 \\ \alpha_{\text{down}} & \text{if } \Delta P_{\text{Tot}}(t_k) > 0 \end{cases} \\ P_{Tot_a}(t_k) &= P_{Tot_a}(t_{k-1})(1 - \alpha_{pw}(t_k)) \; + \; \Delta P_{\text{Tot}}(t_k)\alpha_{pw}(t_k) \end{aligned} \tag{4.13}$$

where α_{up} and α_{down} are two EMA parameters indicating whether the estimation is greater or lower than the measured value. This configuration allows to define a faster response when the estimation is higher and a more balanced response when it is lower. Additionally, the α_* parameters can change in time depending on how long the estimation has been above or below the measured value. Notably, the parameter d in (4.13) accounts for the controller delay discussed in section 3.3.4; the $k-1$ time step of P_{Mes} refers to the measurement being taken during the previous controller sampling period t_s.

The function $g_{\mathcal{D}}$ responsible for selecting the voltage of the domain follows the state-of-the-art approach of choosing the maximum selected voltage to avoid cutting the performance of the highest-demanding PE [12]: $g_{\mathcal{D}}(V_i) = \max(V_i)$. However, this is a naive method that does not account for optimizations aimed at minimizing (3.1) according to the 'importance matrix' $\mathcal{R}$, thermal evolutions and predictions, non-linear relations, and couplings.

Despite these enhancements, the EBA approach remains too simplistic to comprehensively address the power and thermal management challenges in many-core systems, particularly under the influence of non-idealities. The following section introduces an alternative algorithm designed to handle these complexities more effectively.

4.3 Fuzzy and Iterative Roots-Finding Method

Rather than applying the coupling constraints before the control process, the Fuzzy-inspired Iterative Control Algorithm (FCA) integrates constraint enforcement directly into the final Conv2F step. This approach uses an iterative roots-finding technique to solve the multi-variable conversion problem within Conv2F. It can be thought of as moving the iterative moving average process, which typically improves over several control intervals t_k, into a single control cycle by employing an iterative solving method. This enables FCA to account for voltage coupling and other constraints faster and more effectively.

The TH-PID is replaced by a fuzzy-inspired LUT control, which offers improved thermal regulation by considering the thermal derivative and directly adjusting frequency and voltage setpoints. Additionally, the PW-DIST mechanism has been improved to account for voltage domain constraints, ensuring better power distribution across shared voltage domains.

4.3.1 Iterative Conv2F Step

For each domain $\mathcal{D}_j$, the relationships (4.6) regarding its PEs are grouped into a system of equations where the domain voltage $V_{\mathcal{D}_j}$ is substituted to V. This ensures

that the voltage coupling is directly enforced in the conversion step:

$$\begin{cases} P_{\mathrm{T},1} = \eta_{\mathrm{est}}\left(F_1,\, V_{\mathcal{D}_j},\, \cdots\right) \\ \quad \vdots \\ P_{\mathrm{T},n_j} = \eta_{\mathrm{est}}\left(F_{n_j},\, V_{\mathcal{D}_j},\, \cdots\right) \end{cases} \tag{4.14}$$

$V_{\mathcal{D}_j}$ is obtained from the coupling function $g_{\mathcal{D}_j}(V_i)$ as:

$$V_{\mathcal{D}_j} = g_{\mathcal{D}_j}(V_{i\in\mathcal{D}_j}) = g_{\mathcal{D}_j}\left(f_V(F_{i\in\mathcal{D}_j})\right) \tag{4.15}$$

Several strategies could be employed to select the voltage level for the domain. To remain consistent with state-of-the-art methods, the maximum selected voltage is chosen, $g_{\mathcal{D}_j}(V_i) = \max(V_i)$. Since $f_V(F_i)$ is a step-wise monotonically increasing function, $V_{\mathcal{D}_j}$ can be rewritten as:

$$V_{\mathcal{D}_j} = f_V\left(\max(F_{i\in\mathcal{D}_j})\right) \tag{4.16}$$

By substituting eq. (4.16) into (4.14), j sets of non-linear systems of equations are obtained, one for each domain $\mathcal{D}_j$:

$$\begin{cases} P_{\mathrm{T},1} = \eta_{\mathrm{est}}\left(F_1,\, f_V\left(\max(F_{i\in\mathcal{D}_j})\right),\, \cdots\right) \\ \quad \vdots \\ P_{\mathrm{T},n_j} = \eta_{\mathrm{est}}\left(F_{n_j},\, f_V\left(\max(F_{i\in\mathcal{D}_j})\right),\, \cdots\right) \end{cases} \tag{4.17}$$

Assuming $\forall\ i = 1, \ldots, n_{\mathcal{D}_j}$

$$\begin{aligned} P_{\mathrm{T},i} &> \min\left[\eta_{\mathrm{est}}(F^S_{\min},\, V_{\mathcal{D}_j},\, \cdots)\right] \\ P_{\mathrm{T},i} &< \max\left[\eta_{\mathrm{est}}(F^S_{\max},\, V_{\mathcal{D}_j},\, \cdots)\right] \end{aligned} \tag{4.18}$$

a solution to the system of equations (4.14) is guaranteed to exist.

As a result, the Conv2F step becomes a root-finding task for j systems of non-linear equations. The global Newton-Raphson iterative method could be employed to solve these systems, using the Softmax function $\mathcal{S}_\alpha$ [60] as a differentiable approximation of the max() function. However, even with hardware-optimized algorithms, solving the global Newton-Raphson method with a reasonable tolerance is not feasible for a real-time control solution running on an embedded LLC with tight timing constraints such as the one described in [110].

Considering that the precision of system actuators is typically limited to $5 \cdot 10^{-3}$, the bisection method [40] presents a suitable alternative to the Newton-Raphson method. While the bisection requires more iterations to converge, each iteration is faster as it does not require inverting the Jacobian, making its execution time less dependent on the dimension of the domain. Additionally, since bisection does not rely on derivatives, the max() function can be used, bypassing the need for the more computationally expensive Softmax approximation. From a deployment perspective on an embedded LLC, this means reducing the execution time, hence being able to execute within the control period.

To further simplify the computations, the function f_V can be upper-bounded by a non-linear polynomial, such as a second-order function:

$$f_V(F_i) = F_i^2 + k_1 F_i + k_0 \tag{4.19}$$

where the coefficients k_1 and k_0 can be determined based on the frequency-voltage relationship of (3.2).

In this improved Conv2F step, both F and V are still treated as continuous variables rather than being quantized to specific levels. This approximation may lead to some performance loss when rounding to the nearest lower level. An alternative approach could involve treating the quantized F and V values by employing control methods for systems over a finite alphabet [152], in which the system operates over discrete levels, or multi-branch tree control schemes [4].

4.3.2 Fuzzy-inspired Thermal Control

The precise operating points provided by the iterative root-finding method, in combination with the EBA PID led to significant oscillation yielding reduced control performance. Although the thermal evolution of the system can be described by a linear relation, the non-linearities and couplings introduced by real-world conditions expose the limitations of the PID as a linear time-invariant controller.

In particular, the discrete coupled outputs which are exponentially related to the temperature state, require PID tunings geared toward fast response to prevent the thermal runaway. However, such tunings also lead to high output oscillations, resulting in control performance degradation. These oscillations were especially evident when the PID was used in conjunction with the improved Conv2F presented in section 4.3.1. Consequently, the thermal capping regulation (TH-PID) was replaced by a heuristic, LUT-based control inspired by fuzzy control theory.

Fuzzy control theory provides a simple and effective way to translate symbolic decision tables into control laws. In practice, fuzzy logic involves three key steps [106]:

1. Control inputs are converted into fuzzy sets through a process known as ***Fuzzification***
2. Logic rules are applied to determine the appropriate control signal
3. The control signal is converted back into a quantitative output in a process called ***Defuzzification***

Fuzzy control is applicable to non-linear systems as it does not require a model, is simple to compute and modify, and offers modular flexibility. The logic set of rules can be approximated into a LUT or by a mathematical function for faster computation of the control actions.

Fuzzy logic's capability to handle non-linearities allows the thermal control to be applied directly to the frequency F and voltage V values. For this reason, the thermal capping step in the FCA structure is placed before the power distribution step. This modification ensures that power already allocated to PEs is not lost by subsequent reductions caused by thermal constraints, improving effective power utilization. The original BA structure was based on the assumption that the power directly influences the temperature. However, with the updated model, temperature also affects power, making the previous control strategy less effective.

In FCA, a *fuzzy state* is assigned to each PE. This fuzzy state embodies the reduction in frequency and voltage due to thermal regulation. A LUT is constructed by dividing

ΔT/T[°C]	< 45	$45 - 65$	$65 - 80$	$80 - T_L$	$\geq T_L$
< 0	2	2	1	0	-1
$0 - 0.5$	2	1	1	0	-1
$0.5 - 1.0$	1	1	0	-1	-2
$1.0 - 2.0$	1	0	-1	-2	-3
≥ 2.0	0	0	-2	-3	-4

TABLE 4.1: LUT table used to determine the increment or decrement of the fuzzy state based on the temperature derivative $\Delta T(t_k)$ and the temperature $T(t_k)$.

the temperature variable of each PE $T_i(t_k)$ and its derivative $\Delta T_i(t_k)$, into regions relevant to thermal control. The LUT is then populated with values that determine whether to increase or decrease the fuzzy state.

The derivative is approximated as the difference between the current temperature and the temperature of the previous s-th steps $\Delta T(t_k) = T(t_k) - T(t_k - s)$, to help filter sensors' noise. The choice of s depends on the ratio between the system's fastest thermal time constant and the execution periodicity of the thermal step t_k, as well as the ratio between the maximum potential temperature variation in a t_k and the amplitude of the noise.

The number of columns and rows in the LUT is determined primarily by the memory and computational constraints of the LLC, and secondarily by the desired quality of thermal control. The selection of the derivative threshold values is largely based on the thermal time constants of the system, while the temperature thresholds depend mainly on the slope of the power-temperature leakage relationship (see Figure 3.2). The LUT values can be defined by discretizing a surface that represents a trade-off between performance reduction and control effectiveness. The specific LUT used in this work is shown in table 4.1.

Each negative integer value of the fuzzy state reduces half of the frequency step between two consecutive voltage levels, with every even negative fuzzy state value resulting in a one-level reduction in voltage. This configuration can also support a frequency turbo boost feature through positive fuzzy states. However, since turbo capabilities are not analyzed in this work, the fuzzy states are saturated at the upper boundary of 0 to ensure a fair comparison with other algorithms.

4.3.3 PW-DIST and Conv2F Improvements

Two additional modifications are introduced to improve the other steps of the FCA algorithm. First, the PW-DIST step is updated to account for the voltage domain coupling, improving the allocation of power. This is an important addition as all considerations related to domain coupling are deferred to the final conversion step. Second, a hysteresis low-pass filter is applied after the Conv2F step, designed to reduce oscillations caused by rapid voltage fluctuations. These refinements improve efficient power utilization and control signal smoothness, particularly in scenarios when both the thermal and power control are engaged.

The power distribution step is updated to account for voltage-sharing domain configurations. Instead of applying the α power distribution algorithm (4.5) to each individual PE, as the BA, it is now applied at the domain level. A linear combination of the maximum and average temperature of the PEs within each domain, along with the total estimated power of the domain $\sum_i^{n_{\mathcal{D}_j}} P_{\text{est},i}$, serve as the inputs to the α

heuristic algorithm. Defining the domain temperature $T_{\mathcal{D}_j}$ as:

$$T_{\mathcal{D}_j} = k_{T1} \max(T_i) \; + \; k_{T2} \frac{\sum_i^{n_{\mathcal{D}_j}} T_i}{n_{\mathcal{D}_j}} \qquad \forall \, i \in \mathcal{D}_j \tag{4.20}$$

where k_{T1} and k_{T2} are the combinatorial coefficients, the α heuristic algorithm becomes:

$$\begin{aligned} \alpha_j &= \frac{1}{T_{\mathrm{L,i}} - T_{\mathcal{D}_j}} \qquad \forall \, j \in n_d \\ \bar{\alpha}_j &= \frac{\alpha_j}{\sum_{j=1}^{n_d} \alpha_j} \\ P_{\mathcal{D}_j} &= \sum_i^{n_{\mathcal{D}_j}} P_{\mathrm{est},i} - \bar{\alpha}_j \Delta P_{\mathrm{C}} \end{aligned} \tag{4.21}$$

where $P_{\mathcal{D}_j}$ is the power allocated to the j-th domain.

Within each domain, the power is distributed based on the workload ω_i of each PE, prioritizing the ones executing high-performance workloads. Defining the average C_{eff} value within each domain $\mathcal{D}_j$ as:

$$\zeta_{\mathcal{D}_j} = \frac{\sum_i^{n_{\mathcal{D}_j}} C_{eff,i}}{n_{\mathcal{D}_j}} \qquad \forall \, i \in \mathcal{D}_j \tag{4.22}$$

the heuristic power distribution algorithm within each j-th domain becomes:

$$\begin{aligned} \alpha_i &= \frac{1}{C_{eff,i} + \zeta_{\mathcal{D}_j}} \qquad \forall \, i \in \mathcal{D}_j \\ \bar{\alpha}_i &= \frac{\alpha_i}{\sum_{i=1}^{n_{\mathcal{D}_j}} \alpha_i} \\ P_{\mathrm{T},i} &= P_{\mathrm{est},i} - \bar{\alpha}_i \big(\sum_i^{n_{\mathcal{D}_j}} P_{\mathrm{est},i} - P_{\mathcal{D}_j} \big) \end{aligned} \tag{4.23}$$

The $\zeta_{\mathcal{D}_j}$ term is introduced to achieve a more balanced power distribution, ensuring that mid- and low-power workloads are not overly restricted.

The hysteresis filter is implemented to mitigate frequent voltage changes resulting from the iterative root finding, which determines precise operating points at each iteration t_k. These adjustments, often by a single level, can induce oscillations and lead to excessive hardware utilization. The filter delays voltage increases by ρ steps, while allowing immediate voltage reductions, ensuring that power and thermal capping regulation remain unaffected.

For each domain, the filter monitors whether $V_{\mathrm{new},j}$, the updated voltage proposed by the FCA algorithm, remains consistently higher than the current voltage $V_{a,j}$. If this condition holds after ρ iterations, the final voltage at t_k is applied. The process can be described as follows:

$$V_a(t_k) = \begin{cases} V_{\mathrm{new}}(t_k), & \text{if } V_{\mathrm{new}}(t_k) < V_a(t_{k-1}) \\ V_{\mathrm{new}}(t_k), & \text{if } V_{\mathrm{new}}(t_i) > V_a(t_i) \quad \forall \, i \in [k, k-\rho] \\ V_a(t_{k-1}), & \text{otherwise} \end{cases} \tag{4.24}$$

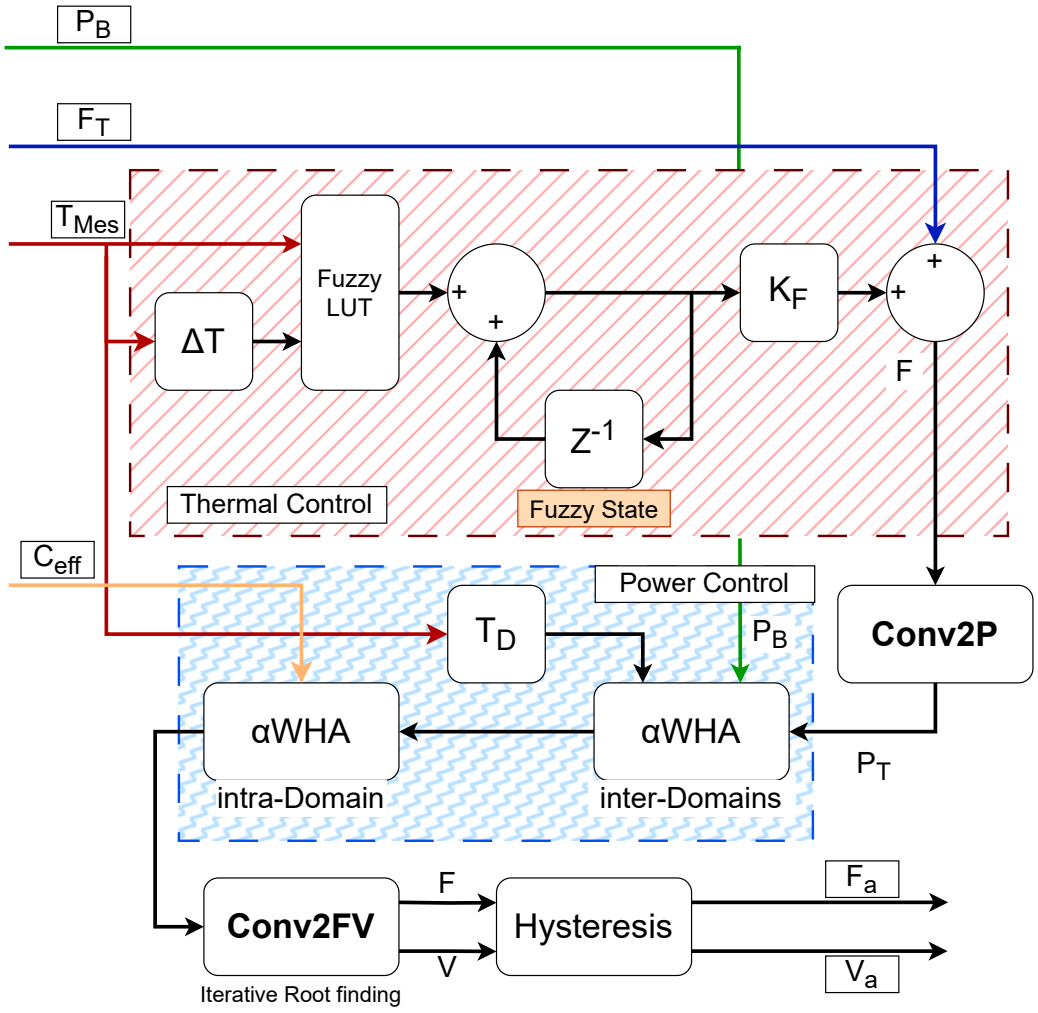


FIGURE 4.2: The whole structure of the Fuzzy-inspired Iterative Control Algorithm. P_B, F_T, T_i, and C_{eff} are the inputs, while F_a and V_a are the resulting outputs.

Determining the appropriate value of ρ is essential for achieving a balance between smoothing voltage fluctuations and ensuring an effective controller response.

The whole structure of the FCA is shown in Figure 4.2.

4.4 Model Predictive Control

Model Predictive Control (MPC) offers a powerful framework for optimizing control actions in dynamic systems, particularly when dealing with several constraints and multi-variable control objectives [61]. In the context of HPC, MPC can be leveraged to predict future system states and adjust control signals to meet both power and thermal management constraints, while optimizing for competing objectives, such as performance and energy efficiency.

MPC relies on dynamic models of the system to predict its future behavior over a finite time horizon. At each control step, it solves an optimization problem to determine the optimal control signals that minimize a given cost function J while satisfying system constraints. Despite its computational complexity, it is widely used in applications requiring precise control of dynamic systems, and its characteristics

fully align with the control problem described in eq. (3.1), making it a suitable option for DVFS control and DPTM in HPC systems [50].

Compared to previously discussed algorithms, MPC requires an accurate model of the system, and, depending on the available sensors, it may also require the use of a state observer to estimate unmeasured states. This adds a layer of complexity, as the controller must account for uncertainties both in the model and in the state estimation. Additionally, MPC is computationally more demanding, making its implementation in real-time controllers with limited processing capabilities and tight time constraints, such as those used in HPC environments [110], a significant challenge.

Additional considerations must be made due to the shared nature of the power budget across multiple PEs and subsystems. Effective management of power resource distribution requires either a centralized control approach or a relaxation of constraints, as necessitated by a distributed control framework (see section 4.5 for details). However, centralized MPC does not scale well, leading to prohibitive computational requirements as the number of PEs increases. To address the scalability issue, the MPC could be divided into smaller problems [23]. The optimization problem is typically formulated as a Quadratic Programming (QP), which scales polynomially with the number of decision variables n_v, constraints n_c, and the length of the prediction horizon N_{hzn}. The typical time complexity for QP solvers is $\mathcal{O}(n^3)$ per iteration, where $n = N_{hzn}(n_v + n_c)$ [144]. As established in Lemma 4.4.1, distributing the MPC control problem among PEs by relaxing power budget constraints leads to a consistent reduction in computational complexity, with the best scenario arising when each PE is considered individually.

Lemma 4.4.1. *Let $n \in (1, \infty)$ be a natural number, and let $b_i \in \mathbb{N}^+$ for all $i \in \{1, \ldots, n\}$. Then, the following inequality holds:*

$$\sum_{i=1}^{n} b_i^3 \; < \; \left(\sum_{i=1}^{n} b_i \right)^3$$

Proof. We begin by expanding the right-hand side of the inequality:

$$\left(\sum_{i=1}^{n} b_i \right)^3 = (b_1 + b_2 + \cdots + b_n)^3 .$$

Using the binomial theorem for expanding powers of a sum, we get:

$$\left(\sum_{i=1}^{n} b_i \right)^3 = \sum_{i=1}^{n} b_i^3 + \text{cross-terms}.$$

The cross-terms consist of all products of the form $3b_i^2 b_j$, $3b_i b_j^2$, and $6b_i b_j b_k$ (for distinct i, j, k).

Since each $b_i > 0$, all the cross-terms are strictly positive. Therefore, we conclude that:

$$\sum_{i=1}^{n} b_i^3 < \left(\sum_{i=1}^{n} b_i \right)^3 .$$

In other words, the sum of the cubes of the b_i's is strictly less than the cube of the sum, due to the contribution of the positive cross-terms. □

Furthermore, the described control problem is inherently non-linear and involves hybrid coupling constraints where voltage depends on frequency and is shared across domains. Solving such non-linear and hybrid optimization problems in real-time exceeds the capabilities of microcontrollers that are feasibly implementable within an HPC chiplet [137]. Approximations must be introduced to deal with the non-linear and hybrid nature of the problem.

In this work, the centralized MPC is employed, as implementing the distributed approach would probably cause power constraint violation and require a further analysis into the degree of relaxation that could be tolerated. Using advanced computational methods [144, 170, 109], the linear MPC variant can be feasibly implemented on the TPC, provided that t_s and N_{hzn} are appropriately chosen. Nonetheless, the distributed scenario remains an area of interest, particularly as industry is moving toward integrating small, lower-capability local controllers at each PE, primarily for managing local temperature control (2.3.1).

4.4.1 Control Problem Layout

An important initial decision is the choice of the control input. Selecting power consumption P_T ensures a linear MPC problem, as it reduces the system model to the thermal dynamics alone. However, this approach prevents the optimization from directly considering frequency and voltage separated from ω, necessitating the Conv2P and Conv2F steps and thus reintroducing the challenges discussed in previous sections. Selecting Frequency F, Voltage V, or both as control inputs is not enough to linearize the system, even modelling V as a linear function of F, as the dynamic power eq. (2.8) includes the term FV^2.

To address this nonlinearity, the chosen control input is $P_M = FV^2$. This ensures that the dynamic power remains linear in the MPC model while providing the capability to directly optimize on F and V. The temperature T is the MPC state for the system model. However, the leakage power P_stat still remains a non-linear term even with the choice of P_M, thus requiring linearization. To achieve a linear MPC formulation, P_stat is approximated through a linear combination of the state and input variables as follows:

$$P_\mathrm{stat} = k_1 T + k_2 P_M + k_0 \tag{4.25}$$

Using the thermal model (2.6) and the power model (2.8), the linearized system model for the MPC is formulated as:

$$\dot{T} = AT + B(P_M \omega + k_1 T + k_2 P_M + k_0) \tag{4.26}$$

The constraints are taken from the control problem formulation (3.1):

$$\begin{cases} T_i \leq T_\mathrm{L}, & i = 1, \ldots, n_c \\ \sum_{i=1}^{n_c} \left(k_1 T_i + P_{M,i}(\omega_i + k_2) + k_0\right) \leq P_\mathrm{B}(t) \\ \sum_{i=1}^{n_{\mathcal{D}_j}} \left(k_1 T_i + P_{M,i}(\omega_i + k_2) + k_0\right) \leq P_{\mathcal{D}_j}(t), & i \in \mathcal{D}_j, \quad j = 1, \ldots, n_d \end{cases} \tag{4.27}$$

Additional constraints, such as lower and upper bounds on P_M and the linearized power $(P_M \omega + k_1 T + k_2 P_M + k_0)$, can be added effortlessly.

The initial optimization cost J is derived from the primary control objective (3.1) of minimizing the difference between the target and actual frequency. Additional penalization terms are introduced for both the state and input variables, which are normalized to their respective range values to ensure consistency during the optimization process. The cost function is designed to be convex, ensuring that the

optimization problem remains solvable efficiently.

$$J = (\hat{P_M^*} - \hat{P_M})^T \boldsymbol{R_t} (\hat{P_M^*} - \hat{P_M}) + \hat{P_M^*}^T \boldsymbol{R} \hat{P_M} + \hat{T}^T \boldsymbol{Q} \hat{T} \tag{4.28}$$

All variables are treated as a vector, i.e. $T = [T_1, \cdots, T_{n_c}]$, and the hat notation indicates that the variable is normalized. $P_M^* = F_T * f_V(F_T)^2$ represents the given target. $\boldsymbol{R_t}$, $\boldsymbol{R}$, and $\boldsymbol{Q}$ are positive definite symmetric matrices that contain the optimization coefficients.

Due to the choice of P_M as the control input, which separates it from the workload ω, individual weights per PE can be incorporated into the control input penalization matrix R to prioritize certain PEs over others for performance reasons. Had the target power consumption P_T been used instead, adding a performance weight would have penalized not only the dynamic power but also the static power component. Additionally, the dynamic power would have been subject to a scaling factor dependent on the currently executed workload.

Let $\theta_i \in [0, 1]$ represents the weight of each PE indicating its relative importance, with the goal of minimizing its penalization in the optimization process. Let $\theta' = 1 - \theta$ represent the complementary weight, the cost function can be reformulated as:

$$J = \big((\hat{P_M^*} - \hat{P_M}) \cdot \theta'\big)^T \boldsymbol{R_t} \big((\hat{P_M^*} - \hat{P_M}) \cdot \theta'\big) + (\hat{P_M^*} \cdot \theta')^T \boldsymbol{R} (\hat{P_M^*} \cdot \theta') + \hat{T}^T \boldsymbol{Q} \hat{T} \tag{4.29}$$

where θ' is the vector collecting all the θ_i' weights, and $\cdot$ denotes the scalar product.

The incorporation of the power constraints in eq. (4.27) and the theta adjustment in (4.29), introduce an advanced power allocation capability within the MPC controller. This represents a novel approach, as MPC is typically applied solely as a thermal control mechanism, as highlighted in section 4.1. Integrating power regulation alongside thermal regulation enhances the synergy between these control functions, potentially yielding an improved operating point.

Additional considerations must be made when selecting the time horizon N_{hzn}, as it significantly affects both the performance and computational feasibility of the MPC. Longer time horizons enable the controller to better anticipate and manage future system dynamics; however, this comes at the expense of increased computational complexity, which may exceed real-time control constraints. In the described HPC system, future predictions are still prone to inaccuracies and significant deviations due to the variability and unpredictability of the workload ω as discussed in section 3.3.3. Therefore, choosing a large time horizon is unlikely to yield significant, if any, improvements.

A balanced approach is to select the time horizon such that the MPC prediction covers a period between 1 and 2 times the fastest thermal time constant t_{Th}. Therefore, based on the controller execution interval t_s, the time horizon can be defined as:

$$N_{hzn} = \frac{\lambda \cdot t_{\text{Th}}}{t_s} \qquad \text{with } \lambda \in [1, 2] \tag{4.30}$$

This leads to important considerations regarding the choice of t_s for the MPC controller. Given the significant computational complexity of the centralized MPC in systems with a large number of PEs, which scales with the choice of N_{hzn}, selecting a larger t_s not only provides more time for the MPC solver to complete its computation, it also reduces the value of N_{hzn} according to eq. (4.30), further reducing computational requirements. On the other hand, the choice of t_s must still account for the fast system dynamics and the exponential behavior described in section 3.3.1.

Relying on the quality of the MPC control and prediction, a sampling interval of $t_s = 1ms$ was selected in this work. While this is larger than the intervals used by other presented control algorithms, it remains within a reasonable range. Faster fluctuations can still be addressed effectively through the use of a separate high-frequency power mitigation loop, as described in section 3.3.4.

4.4.2 Linearization

To linearize the static power P_{stat}, the least-squares method was employed. This approach minimizes the sum of the squared differences between the model's predicted values and the actual data [33]. By fitting the linear polynomial formulation of P_{stat} (4.25) to the nonlinear model, the least-squares method ensures that the linearization best approximates the underlying trend of the data while minimizing the error across all points.

Let $M = [a, b] \times [c, d]$ be the domain of the leakage power P_s [1], where $T \in [a, b]$ and $V \in [c, d]$. The objective is to minimize the square of the L_2-norm of the approximation error $S()$ over the set of parameters in $\mathbb{R}^3$:

$$S(k_1, k_2, k_0) = \int_M \left(P_s(T, V) - (k_1 T + k_2 P_M(V) + k_0)\right)^2 dT\, dV \tag{4.31}$$

where k_0, k_1, and k_2 are the parameters of the linearized model to be determined. Denoting the L_2-inner product as:

$$\langle g, h\rangle = \int_M g(x, y)h(x, y)dxdy \tag{4.32}$$

the minimum is found with:

$$\begin{cases} 0 = \frac{1}{2}\partial S/\partial_{k_0} = k_0\langle 1, 1\rangle + k_1\langle 1, T\rangle + k_2\langle 1, P_M\rangle - \langle 1, P_s\rangle, \\ 0 = \frac{1}{2}\partial S/\partial_{k_1} = k_0\langle T, 1\rangle + k_1\langle T, T\rangle + k_2\langle T, P_M\rangle - \langle T, P_s\rangle, \\ 0 = \frac{1}{2}\partial S/\partial_{k_2} = k_0\langle P_M, 1\rangle + k_1\langle P_M, T\rangle + k_2\langle P_M, P_M\rangle - \langle P_M, P_s\rangle. \end{cases} \tag{4.33}$$

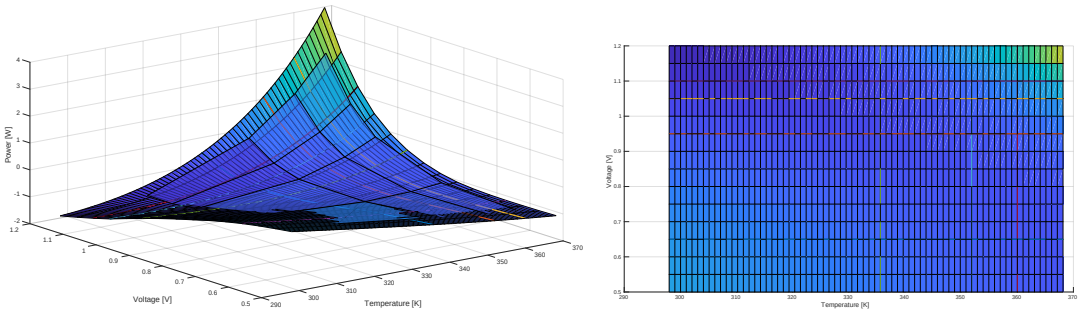
Since the basis $[T, P_M, 1]$ is not orthogonal with respect to the L_2-norm inner product, it is recommended to orthogonalize the basis with Gram-Schmidt [156]. Using:

$$\begin{aligned} \langle 1, 1\rangle &= (b - a)(c - d) \\ \langle 1, T\rangle &= (b^2 - a^2)(d - c)/2 \end{aligned} \tag{4.34}$$

the pair $[1, T - (b + a)/2]$ forms an orthogonal basis in the linear span of 1 and T. By orthogonalizing P_M with respect to this new basis, the third component $P_M - (d + c)/2$ is found. The fact that $\langle P_M, T - (b + a)/2\rangle = 0$ is a coincidence due to the relationship between the basis and the domain of integration. The system of equations (4.33) can be rewritten in the new basis with the coefficients m_i as:

$$\begin{cases} 0 = \frac{1}{2}\partial S/\partial_{m_0} = m_0\langle 1, 1\rangle - \langle 1, P_s\rangle, \\ 0 = \frac{1}{2}\partial S/\partial_{m_1} = m_1\langle T - (b + a)/2, T - (b + a)/2\rangle - \langle T - (b + a)/2, P_s\rangle, \\ 0 = \frac{1}{2}\partial S/\partial_{m_2} = m_2\langle P_M - (d + c)/2, P_M - (d + c)/2\rangle - \langle P_M - (d + c)/2, P_s\rangle \end{cases} \tag{4.35}$$

[1] From this point forward, the static (or leakage) power P_{stat} will be referred to simply as P_s for brevity.



(A) (B)

FIGURE 4.3: Figure 4.3a (left) illustrates the difference between the linearized and actual leakage power models with respect to temperature (T) and voltage (V). The overlaid surface is interpolated among the selected k_{res} values that minimize variance within each region. Colored lines indicate the division of the original surface into those regions. Figure 4.3b (right) provides a top-down view, making the regional segmentation clearer.

Solving the system of equations (4.35) with respect to the m_i coefficients yield:

$$\begin{aligned} m_0 &= \frac{\langle 1, P_s \rangle}{\langle 1, 1 \rangle} \\ m_1 &= \frac{\langle T - (b+a)/2,\ P_s \rangle}{\langle T - (b+a)/2,\ T - (b+a)/2 \rangle} \\ m_2 &= \frac{\langle P_M - (d+c)/2,\ P_s \rangle}{\langle P_M - (d+c)/2,\ P_M - (d+c)/2 \rangle} \end{aligned} \tag{4.36}$$

and the linearized model for P_s is given by:

$$\begin{aligned} P_s &\approx m_1(T - (b+a)/2) + m_2(P_M - (d+c)/2) + m_0 \\ &= m_1 T + m_2 P_M + (m_0 - \frac{m1(b+a) + m_2(d+c)}{2}) \end{aligned} \tag{4.37}$$

From (4.37), the k_i coefficients are derived as follows:

$$\begin{aligned} k_1 &= m_1 \\ k_2 &= m_2 \\ k_0 &= (m_0 - \frac{m_1(b+a) + m_2(d+c)}{2}) \end{aligned} \tag{4.38}$$

To correctly integrate P_M with respect to V, it is necessary to express F as a continuous function of V. This can be achieved by inverting the second-order polynomial approximation of the f_V relation (4.19), as:

$$F = f_V^{-1}(V) = c_1 + (V + c_2)^{\frac{1}{2}} \tag{4.39}$$

As a result, the P_M used to compute the k_i coefficients is expressed as:

$$P_M = (c_1 + (V + c_2)^{\frac{1}{2}})V^2 \tag{4.40}$$

4.4.3 Time-varying Linearization Offset

Linearizing the static power P_s still results in discrepancies between the approximated model and the real system behavior. This difference is especially present in high-temperature and high-voltage regions, where the non-linearity of P_s is significantly pronounced and where the control is primarily engaged. The difference between the power model and linearization used in this work is shown in Figure 4.3a. Attempting to refine the linearization to better approximate these areas could lead to either overly conservative results or, in some cases, even negative values due to the steep slope of the curve in those regions. To address this issue, an additional input k_{res} is introduced into the MPC model.

k_{res} is a time-varying residual adjustment for each PE to the offset term in eq. (4.25). The term is added at each iteration of the controller, by determining the difference between the original leakage power model P_s and its linear approximation. This difference, denoted as $P_{\Delta s}$, represents the error introduced by the linearization. The 3D surface of $P_{\Delta s}$ is discretized into a LUT that depends on the current voltage V and temperature T values. The rationale for employing a LUT, rather than directly computing the exact $P_{\Delta s}$ value, lies in the benefits of discretizing the surface into regions with minimal variance, which may improve the overall approximation accuracy and computational efficiency.

Indeed, the inclusion of k_{res} introduces an inherent approximation, as it is treated as a fixed value during the MPC optimization process. Throughout this optimization, the MPC outputs may converge to operating points that differ significantly from those used in the computation of k_{res}, potentially reducing its effectiveness. By providing a less precise initial value but one that is regionally more accurate, the MPC will provide more accurate optimization results in that region.

To ensure proper discretization into the LUT, the surface must be divided into rectangular regions with parallel axes where adjacent regions share common boundaries. An algorithm is employed to divide the surface in such a way that these regions are well-suited for table representation, while also minimizing the sum of the variance within each region. This approach ensures a more precise and uniform approximation of the operating points.

The selection of the n_V and n_T divisions for the V and T ranges respectively, should account for both memory constraints and the need to create regions large enough to meet the aforementioned criteria. An example of the resulting region division with $n_V = 4$ and $n_T = 3$ is illustrated in Figure 4.3b. The formulation of the MPC remains unchanged, as it is sufficient to substitute the parameter k_0 with $k_0^+ = k_0 + k_{\text{res}}(t)$.

4.4.4 Hybrid Integration

The hybrid nature of the control problem originates from the relationship between F and V described in eq. (3.2). This dependency is not continuous, but rather it divides the operating space into feasible regions. Since the chosen control variable is $P_M = FV^2$, and the F-V relationship has been approximated using continuous functions, this hybrid characteristic cannot be directly implemented in the current MPC framework.

To address this issue, a workaround is to guide the MPC optimization by embedding in the cost function a preference for similar P_M values across elements within the same voltage domain. This inclusion encourages PEs to operate at similar points on the F-V curve, thereby avoiding scenarios where PEs within the same voltage domain

have vastly different operating points, which would lead to suboptimal performance, as described in section 3.3.2.

For each domain $\mathcal{D}_l$, the convex term used to penalize large value differences in P_M values across PEs within the domain is given by:

$$k_{ij}(P_{M_i} - P_{M_j})^2 \quad \forall\, i,j \in \mathcal{D}_l \;\text{ with }\; i > j \tag{4.41}$$

where k_{ij} are the penalization terms.

Considering the additional term $P_M^T \mathcal{H} P_M$ in the cost function J (4.28), the quadratic-form matrix $\mathcal{H}$ is constructed as follows. For each domain $\mathcal{D}_l$ and for each pair of distinct PEs $i \neq j$ within $\mathcal{D}_l$, the (i,j)-th element of $\mathcal{H}$, as well as its symmetric (j,i)-th element, are set to $-k_{ij}$. All inter-domain elements, corresponding to pairs $i \neq j$ where $i \in \mathcal{D}_l$ and $j \notin \mathcal{D}_l$, are set to zero. On the main diagonal, the (i,i)-th elements are assigned the negative sum of all other elements in the corresponding row (or equivalently, column). The procedure is summarized as follows:

$$h_{ij} = \begin{cases} -k_{ij}, & \text{if } i,j \in \mathcal{D}_l \;\text{ and }\; i > j \\ -k_{ij}, & \text{if } i,j \in \mathcal{D}_l \;\text{ and }\; j > i \\ 0, & \text{if } i \in \mathcal{D}_l,\; j \notin \mathcal{D}_l \;\text{ and }\; i \neq j \\ \sigma_i, & \text{if } i = j \end{cases} \tag{4.42}$$

with:

$$\sigma_i = h_{ii} = -\sum_{i}^{i \neq j} k_{ij} = -\sum_{j}^{i \neq j} k_{ij} \tag{4.43}$$

To illustrate the selection method for the h_{ij} elements concerning the penalization term in (4.41), consider a simple example involving a single 3x3 domain. The additional term in the cost function is given by:

$$\begin{aligned} u^T \mathcal{H} u &= \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} (k_1+k_2) & -k_1 & -k_2 \\ -k_1 & (k_1+k_3) & -k_3 \\ -k_2 & -k_3 & (k_2+k_3) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \\ &= (k_1+k_2)u_1^2 - 2k_1u_1u_2 - 2k_2u_1u_3 + (k_1+k_3)u_2^2 - 2k_3u_2u_3 + (k_2+k_3)u_3^2 \\ &= k_1(u_1-u_2)^2 + k_2(u_1-u_3)^2 + k_3(u_2-u_3)^2 \end{aligned} \tag{4.44}$$

which is the one defined in (4.41).

The resulting optimization cost is:

$$\begin{aligned} J = \big((\hat{P_M^*} - \hat{P_M}) \cdot \theta'\big)^T \boldsymbol{R_t} \big((\hat{P_M^*} - \hat{P_M}) \cdot \theta'\big) \;+\; \hat{P_M}^T \boldsymbol{\mathcal{H}} \hat{P_M} \\ (\hat{P_M^*} \cdot \theta')^T \boldsymbol{R} (\hat{P_M^*} \cdot \theta') \;+\; \hat{T}^T \boldsymbol{Q} \hat{T} \end{aligned} \tag{4.45}$$

Additional considerations could be made regarding whether the weight term θ' should also be applied to the hybrid cost term.

4.5 Distributed Control

Until now, the presented control approaches were focused on a single computing chiplet. However, in typical HPC systems, several chiplets are distributed across multiple sockets on the same motherboard. This introduces a new layer of interactions

between the different components and the need to develop a control framework that manages power distribution optimally across the entire system.

A similar situation emerges with the interaction among Local Controller for Distributed Controls (LDCs), a new trend in HPC design discussed in section 2.3.1. This configuration requires a coordinated power management strategy across all controllers to ensure optimal performance. LDCs apply the control action on their specific areas, but the overall system performance depends on a higher-level coordination effort to avoid suboptimal global resource allocation.

These control structures fit well within the framework of distributed control methodologies [38]. In distributed systems, controllers operate locally with individual algorithms on a common variable, while abiding by global constraints and exchanging limited information with their neighbors. Additionally, distributed control theories generally do not require full-duplex direct communication between all components, which helps to avoid increased latency and congestion within the NoC.

These principles of distributed control are rooted in consensus-based methods and other techniques from network control theory, where the interactions among agents are modeled using graph theory. Each agent aims to maximize its local utility, often competing for shared resources such as available power. However, global optimization requires these agents to agree on a shared set of values and constraints to prevent conflicts and ensure optimal resource distribution across the entire system [38].

4.5.1 Control Problem for Distributed Entities

Consider a configuration with four controllers arranged in a square, where each component communicates only with its adjacent ones on the horizontal and vertical sides. This setup defines a strongly connected graph [38]. Each of these four controllers manages DPTM for its specific area and is constrained by a shared total power budget. The control problem is to distribute this shared power optimally among the four entities, each with varying requirements and conditions.

For instance, assume that power is initially distributed equally. Then, one controller experiences a reduction in power demand due to thermal capping in its area, another sees reduced consumption as its controlled PEs enter an idle state, while yet another controller requires increased power to support a higher DVFS operating point. The objective of this control problem is to dynamically allocate power among peer controllers in response to these variable demands, optimizing according to a predefined cost function.

The control configurations described above can be recast into a cost-coupled optimization problem [107]. In distributed control systems, a cost-coupled optimization problem arises when several interconnected agents aim to minimize a collective cost function that is the sum of local cost functions, each dependent on a common decision variable. Each agent has only partial information about the global system, typically limited to its own local objective and constraints, and only a small part of data has to be shared with a subset of agents. The objective is to achieve a globally optimal solution while ensuring the overall system constraints are satisfied. The equation describing cost-coupled problems is:

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{N} f_i(x) \quad \text{with } x \in \mathcal{X} \tag{4.46}$$

where $\mathcal{X}$ represents the global constraints set, x the common variable, and f_i the local cost function to optimize. In our particular case, x is the vector of all controllers' power consumptions P_i, $\mathcal{X}$ encapsulates the PDN power boundary limits and the shared power budget limit P_B, and N is the total number of controllers.

The power budget constraint is also integrated into the cost function, alongside the tracking of each controller's specific power consumption target. This allows the cost function to serve as a balancing mechanism between local controller decisions and global system requirements. The local controller's cost function can thus be formulated as:

$$f_i = r_i(P_i - P_i^*)^2 + \frac{(1 - r_i)}{N} \left(\sum_{i=1}^{N} P_i - P_B \right)^2 \tag{4.47}$$

where P_i^* represents the target power consumption selected by the i-th controller for its area, and P_i is the resulting allocated power. The parameter $r_i \in (0, 1)$ is an optimization coefficient specific to each controller, which could be determined as a linear combination of the area's thermal headroom and current workload, thus prioritizing controllers managing high-demand workloads with sufficient thermal headroom.

To enforce the constraint set $\mathcal{X}$, the resulting P_i in each controller are projected onto $\mathcal{X}$ to ensure that any candidate solution remains within feasible bounds. Formally, for a given point x outside of $\mathcal{X}$, the projection $\Pi_{\mathcal{X}}(x)$ is defined as the closest point in $\mathcal{X}$ to x in terms of Euclidian distance. This projection step can be incorporated into the control algorithm, allowing each controller to update its decision variables while remaining compliant with the shared power budget. By doing so, the optimization process respects the feasibility of each controller's power allocation and prevents constraint violations in the distributed system.

4.5.2 The Gradient Tracking Algorithm

The gradient tracking algorithm offers a distributed solution for the cost-coupled optimization problems, such as the one described in section 4.5.1. The algorithm achieves a consensus-based estimate of the global gradient using local limited information exchanges. Compared to other methods, such as the distributed subgradient approach, it achieves a faster convergence rate due to its constant step size [107]. Faster convergence is essential in HPC applications, where rapidly changing conditions and workloads significantly impact power distribution.

In the gradient tracking setup, each controller has its local objective function and operates with partial information, generally restricted to its neighbors' data in a communication network. This limitation is counterbalanced by the algorithm's capability to "track" the gradient across agents dynamically, providing convergence properties close to those achievable by centralized methods.

According to the consensus-based approach, at each time step t_k, each agent i updates its local solution estimate x_i as follows:

$$x_i^{t_k+1} = \sum_{j \in \mathcal{N}_i} a_{ij} x_j^{t_k} - \gamma y_i^{t_k} \tag{4.48}$$

where $\mathcal{N}_i$ denotes the set of neighbors for controller i, γ is the step size, and y_i^t is the dynamically averaged local gradient. y_i^t itself is updated based on local objectives

and neighboring controllers' data as:

$$y_i^{t_k+1} = \sum_{j \in \mathcal{N}_i} a_{ij} y_j^{t_k} + \left(\nabla f_i(x_i^{t_k+1}) - \nabla f_i(x_i^{t_k}) \right) \tag{4.49}$$

ensuring that all local estimates converge asymptotically to the globally optimal solution.

Effective application of the gradient tracking algorithm requires the following conditions [107]:

- A strongly connected communication network among controllers to ensure that local updates propagate throughout the system
- A double-stochastic adjacency matrix to ensure consensus across nodes
- Each local cost function must satisfy the strong convexity property and have Lipschitz-continuous gradients

The weighted adjacency matrix represents the communication links between controllers in a network, where each entry defines the weight of influence one controller has over another. Double-stochasticity implies that the sum of each row and column of the matrix is 1, ensuring that information exchange is balanced and facilitating network-wide convergence [107]. To normalize a non-negative adjacency matrix representing a strongly connected graph, the Sinkhorn–Knopp algorithm can be applied to obtain a double-stochastic matrix [79].

4.6 Conclusion

This chapter has introduced several control algorithms designed to operate power and thermal regulation in HPC systems. Starting from industry-inspired approaches, the discussion has evolved towards more refined and system-aware strategies that better accommodate the complexities of modern computing architectures. Each control method has been developed as a progressive enhancement of the previous one, addressing shortcomings and design pitfalls that lead to poor performance.

The first step in this progression was the Baseline Algorithm (BA), which formalized a structured approach to power and thermal management, reflecting widely adopted industry techniques. This initial design was extended into the Enhanced Baseline Algorithm (EBA), where improvements in voltage regulation and power estimation accuracy were introduced to better align with real-world systems. The Fuzzy-inspired Iterative Control Algorithm (FCA) further advanced this methodology by integrating iterative root-finding techniques for frequency-voltage selection and fuzzy logic for thermal regulation, providing a more robust response to non-linear and coupled system dynamics.

Beyond these iterative refinements, this chapter has explored Model Predictive Control (MPC) approaches, which shift from reactive to predictive strategies, allowing for optimizing the choice of the operating point based on power and thermal model predictions. While MPC introduces computational challenges, its ability to optimize multi-variable constraints and enforce long-term stability represents a promising direction in this field. Finally, the chapter has introduced distributed control formulations, acknowledging the growing complexity of modern architectures where multiple controllers operate concurrently within shared constraints. The discussion of gradient-based optimization methods and consensus algorithms has highlighted

potential pathways for coordinating control decisions across distributed processing units.

Throughout this chapter, each algorithm has been systematically analyzed, with a focus on design motivations, trade-offs, and key implementation challenges. The reasoning behind each refinement has been linked to the findings of Chapter 2 and Chapter 3, ensuring that non-idealities are consistently incorporated into the control strategies. Each algorithm has been designed with real-world deployment feasibility in mind, balancing control effectiveness with execution constraints imposed by embedded microcontrollers in HPC environments. The following Chapter 5 will quantitatively evaluate the performance of these control strategies, comparing their effectiveness under varied workloads, system architectures, and thermal conditions.

Chapter 5

Analysis and Comparison of Control Algorithms

A fair comparison among LLC control algorithms in HPC systems presents significant challenges. The control performance is heavily influenced by the specific microarchitecture of the system. Furthermore, the lack of a standardized metric for comparison presents another significant obstacle, as most benchmarks evaluate the overall processor performance and cannot isolate the controller's contribution. Finally, up-to-date information on state-of-the-art control algorithms is often unavailable or restricted, being guarded by industry confidentiality [16].

Despite these challenges, in this chapter the evaluation of the control algorithms introduced in Chapter 4 is presented, compared to the widely used industry-standard Voting Box Algorithm (VBA) [39, 136, 129]. The IBM On-Chip Controller (OCC) control algorithm from the IBM Power9 chip, publicly released as open-source [73], provides a reproducible baseline for comparison. This industry-standard algorithm is compared to the novel control strategies developed in this work, providing insights into their performance in dynamic power and thermal management.

It is important to note that including HLCs in this comparison would lead to misleading results. As described in chapter 3, HLCs and LLCs are designed to function cooperatively, each targeting specific objectives, execution scopes, and time domains. Although HLCs leverage greater computational resources, they are not suitable for comparison on LLCs objectives, as their longer control step and broader system assumptions limit their effectiveness in handling the fast dynamics that LLCs are optimized for. Moreover, HLCs may lack some features such as applying a given target or jointly enforcing both power and thermal capping, further complicating direct comparisons.

The results presented in this chapter focus on the effectiveness of the proposed control algorithms in managing dynamic workloads and thermal conditions in HPC systems. The analysis covers both steady-state and transient behaviors, with a focus on trade-offs between energy efficiency, thermal stability, and application execution performance. The provided comparison and analysis aims to highlight the strengths and limitations of each algorithm, with a focus on their capacity to manage dynamic system responses and fluctuations, under a wide range of thermal conditions and types of workloads. By examining the achieved trade-offs between performance and effective tracking in different scenarios, this work wishes to provide insights that can guide future improvements in control strategies.

5.1 Evaluation Methodology

To comprehensively assess each control algorithm, their behavior is examined across different system configurations and operating conditions. This includes testing under varied workloads, as well as using different thermal models to simulate diverse cooling environments.

The following three workload scenarios are used to encapsulate the primary operating conditions:

- **MAX-WL:** All PEs execute a vectorized workload with all the data stored in cache, designed to maximize power consumption. The target frequency is set to the maximum value ($3.45GHz$). This scenario is designed to stress the controller's ability to accurately meet the power and thermal constraints, and how effectively it can bring the system to the optimal operating point to meet the given targets under these stringent conditions.

  Additionally, as all PEs operate under identical workloads and frequency targets, and the workload remains constant throughout the test without variations, it provides a reference for assessing the algorithm baseline performance.

- **MULTI-WL:** This scenario is designed to test the controller's ability to effectively allocate the available power across the PEs, each executing different constant workloads with corresponding frequency targets. This configuration is useful to examine how the voltage domain coupling constraints influence the controller's performance with respect to the baseline, and how it can manage simple DVFS targets.

  The PEs are divided into three groups:

  - one executes vectorized workload at maximum frequency ($3.45GHz$)
  - one performs a mix of floating point and integer operations at $2.70GHz$
  - the remaining PEs are kept at the minimum frequency ($0.40GHz$)

  Each voltage domain in each system configuration includes at least one PE from each group to better stress the voltage coupling constraints.

- **CLOUD-WL:** All PEs execute random, dynamically varying workloads, while always requiring the maximum frequency ($3.45GHz$). This scenario is intended to evaluate the controller's responsiveness and adaptability under dynamic and unpredictable workloads. Differently from the previous scenario, in this one the maximum target frequency is always required for all PEs. Controllers able to efficiently allocate power to the most demanding PEs are expected to get a performance advantage.

Each of the workload scenarios is tested using three different thermal model variants to evaluate the adaptability of each control algorithm. These thermal models are designed to reflect common cooling configurations and their impact on silicon temperatures. These configurations are:

- **WATER:** a water-cooled model where PE temperatures are more homogeneous
- **AIR:** an air-cooled model where temperatures follow a Gaussian-shaped spatial distribution across the PEs

- **RACK:** a rack air-cooled model where air flows from the front to the back, creating a column-wise spatial temperature distribution with added thermal coupling among PEs in each column

To fully probe the control algorithm, each test involves adjusting the processor's power budget four times, using both increasing and decreasing patterns. The aim is to capture all possible controller combinations: when only thermal regulation is active, when only power distribution is active, and when both mechanisms are engaged simultaneously. This approach also provides insight into how the algorithms respond to step input changes, particularly in handling both rising and falling signals.

The final stage of evaluation includes running each test under four different voltage domain configurations to assess the effects of voltage coupling on the controller. These configurations are:

- **1-D:** a single voltage domain shared by all PEs. This configuration imposes the most rigid limitations on voltage control, as any adjustments affect all PEs simultaneously, but it could simplify power distribution for some controller compared to a more granular configuration.
- **A-D:** each PE is assigned its own independent voltage domain. This configuration is the simplest, as it eliminates all voltage coupling constraints, reducing the Conv2F to a case where a basic LUT is sufficient to select the optimal TEP operating point.
- **4-D** and **9-D:** these configurations divide the PEs into four and nine voltage domains respectively. These configurations offer progressively less stringent voltage coupling constraints as the number of domains increases, while simultaneously introducing complex challenges in allocating power across multiple domains. Additionally, they facilitate the observation of potential performance trends when comparing the 1-D and A-D configurations.

Parameter uncertainties are introduced into the power and thermal models to simulate real-world variability, while the temperature sensors include $1°C$ white noise to reflect typical PVT sensor specifications. Each test runs for 2 seconds in a Model in the Loop (MIL) framework. The primary comparison involves a 36-core computing chiplet, and is repeated 10 times with varying initial conditions to ensure the results are independent of the system's initial state. For the second series of tests, which involve MPC controllers, a 16-core chiplet is simulated on a set of four different initial conditions. This adjustment is made to reduce simulation time, as the MPC controllers require longer computational times due to quadratic optimization. A single test is shown in Figure 5.1 as an example to illustrate a reference power and thermal evolution.

5.1.1 Metrics and Performance Indicators

The metrics used to assess the performance of the control algorithms across the described batteries of tests, are derived from the control objectives and can be summarized as follows:

- Thermal Regulation:
  - **Peak Temperature Overshoot (TH-MAX)** $[°C]$**:** this metric captures the maximum difference between the silicon temperature with the thermal constraint T_L. The importance of this value lies in the fact that significant

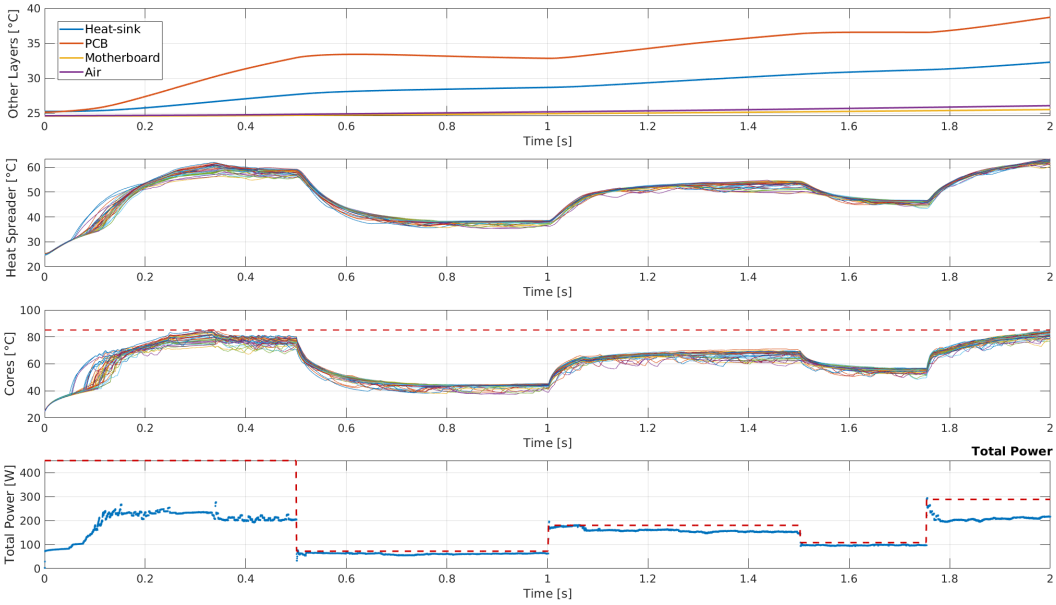


FIGURE 5.1: Thermal and power evolution in the proposed test scenario using the AIR model with a 4-D configuration, executing CLOUD-WL, and controlled by the FCA algorithm. The first three rows illustrate the temperature evolution of thermal dissipation paths (1), heat-spreader (2), and PEs (cores) (3). The last row presents the total measured power consumption. Red dashed lines indicate thermal and power limits.

overshoots can cause system instability, loss of functionality, or permanent hardware damage [158]. TH-MAX allows the comparison of controllers' thermal regulation responsiveness to sudden changes.

- **Cumulative Over-Threshold Time (TH-CT)** [%]**:** this metric is computed as the percentage of time during which the system exceeds the thermal constraint T_L. It only accounts for periods where, in the absence of thermal regulation, the system would exceed the thermal limit, excluding phases with low power budget constraints or reduced frequency targets. This value is used to assess and compare the overall effectiveness of the controller's thermal regulation.

- Power Regulation:

  - **Average Power Overshoot (PW-AV)** [%]**:** this metric is computed as the ratio of the average exceeded power to the power budget target P_B. Differently from the thermal regulation, power spikes are mitigated by the PDN and the capacitances within the power delivery system. However, a sustained average power exceedance significantly above the target can become problematic. PW-AV provides insight into the controller's ability to accurately track the power target.
  - **Cumulative Over-Threshold Time (PW-CT)** [%]**:** this metric is computed as the percentage of time during which the system exceeds the power constraint P_B.

  Both these values can be used to assess and compare the overall effectiveness of the controller's power regulation.

- Target Compliance:
  - **Cumulative Frequency Difference (TF-CFD)** $[GHz/s]$**:** this metric is computed as the L_2-norm of the difference $\Delta F = F_T - F_a$, normalized by the number of elements and control steps $n_s = t_{\text{test}}/t_s$. The normalization ensures a fair comparison between controllers with different execution time intervals t_s. The L_2-norm assumes the $\mathcal{R}$ in (3.1) as the identity matrix, meaning all PEs are considered equally important. Additionally, phases in which the PEs are in an "idle workload" state are excluded from the computation to avoid diluting the results. This metric is important to capture how well LLC controllers achieve their primary performance objective and how optimally they distribute power to PEs over the entirety of the test, for example avoiding over-allocating resources to PEs which would subsequently prompt thermal regulation. Important to note that lower values of TF-CFD indicate better performance.

- Application Execution Performance:
  These supplementary metrics offer an insight into performance objectives that, while not directly managed by the LLCs, are central to the HLC. To ensure they are independent of the specific DVFS target, the metrics are normalized to the baseline results from executing the same workload under unconstrained conditions, without thermal, power, or coupling limitations.
  - **Average Execution Progress (AP-AV)** $[\%]$**:** this metric is computed as the mean of the execution progress across all the considered components. It provides insights into the expected application performance that can be achieved.
  - **Minimum Execution Progress (AP-MIN)** $[\%]$**:** this metric captures the minimum of the execution progress across all the considered components. In addition to indicating the expected worst-case performance, it can be used alongside the AP-AV to analyze how controllers distribute power and penalize different PEs, providing an insight into the range and variation in execution performance.

5.2 Primary Controllers Comparison

The primary comparison aims to establish how the main control algorithms introduced in this work perform relative to state-of-the-art controllers, establishing a baseline for further evaluation. The Fuzzy-inspired Iterative Control Algorithm (FCA) presented in section 4.3 will be compared to the Enhanced Baseline Algorithm (EBA) of section 4.2.1 and the IBM Voting Box Algorithm (VBA), as it was the only current state-of-the-art algorithm with available information [73, 16]. A subsequent second comparison will assess the competitiveness of the MPC controllers under similar conditions, analyzing the innovations introduced in this study.

Results data are aggregated into figures according to workload type (Figure 5.3), Domain number (Figure 5.4), Model type (Figure 5.5), and initial conditions (Figure 5.6). This arrangement provides a structured overview of how each control algorithm responds to each specific configuration and scenario, highlighting performance trends and differences across tests. An additional figure using violin plots (Figure 5.2) illustrates the distribution of results across all tests, offering insights into

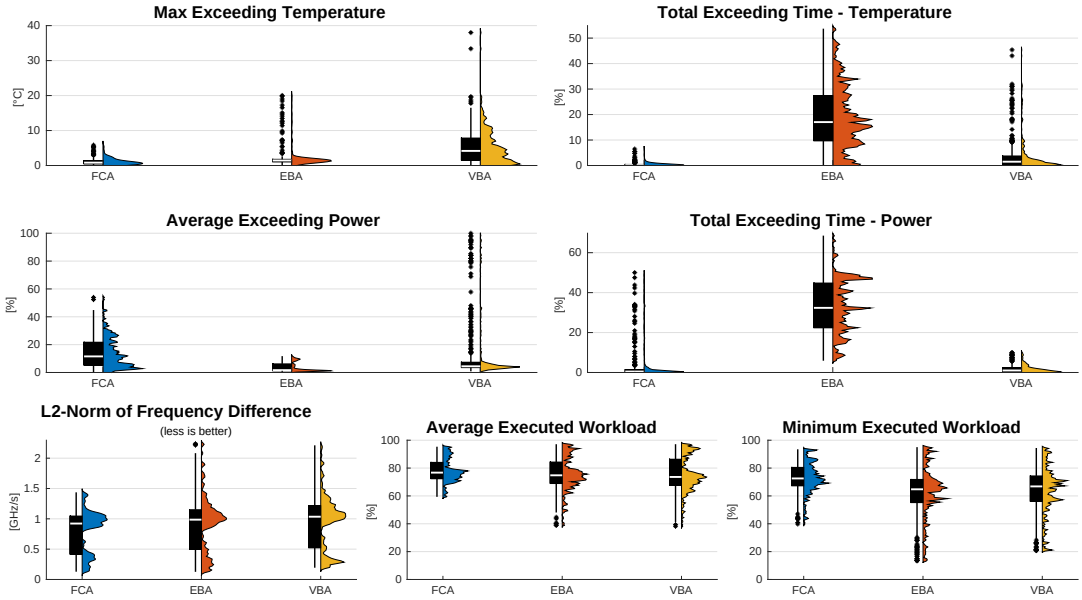


FIGURE 5.2: Overall distribution of control performance metrics for each algorithm. The first row presents thermal metrics, the second row shows power metrics, and the third row displays target compliance and execution progress metrics.

the distribution and consistency of the results. These results were also presented in [16].

5.2.1 Results

Thermal Regulation Metrics

FCA outperforms both EBA and VBA in all thermal capping metrics, exhibiting the least variation among tests. This consistency suggests that the proposed fuzzy-inspired thermal design provides a significant improvement over other PID-based controllers. FCA achieves a TH-MAX maximum temperature that is 2.44 times lower than EBA and 5.52 times lower than VBA, while reducing the average TH-CT by over 90%. These results imply better responsiveness to changes in the underlying dynamics, as clearly shown by the results grouped by workload type in Figure 5.3.

EBA, while showing the highest TH-CT (18.82%), maintains a relatively low TH-MAX across tests, averaging 1.94°C. This implies that reducing the thermal margin by approximately 2°C could effectively eliminate most instances of TH-CT. A notable observation is that EBA's performance improves as the voltage domains' size shrinks (Figure 5.4), likely a reflection of its original BA development without voltage coupling constraints.

In contrast, VBA demonstrates the weakest thermal capping performance of the three algorithms. It produces the highest TH-MAX, posing a risk to hardware safety, and struggles particularly in the RACK model and CLOUD-WL tests, where it records TH-MAX temperatures approaching 20°C, with two spurious results exceeding 30°C. VBA's thermal performance deteriorates as the voltage domains' size increases, indicating significant difficulties in managing voltage coupling.

In particular, both EBA and VBA struggle in the 1-D tests with high initial conditions, likely due to the difficulty in achieving the faster response time required in a fully coupled scenario with reduced heat transmission. This difficulty can be traced

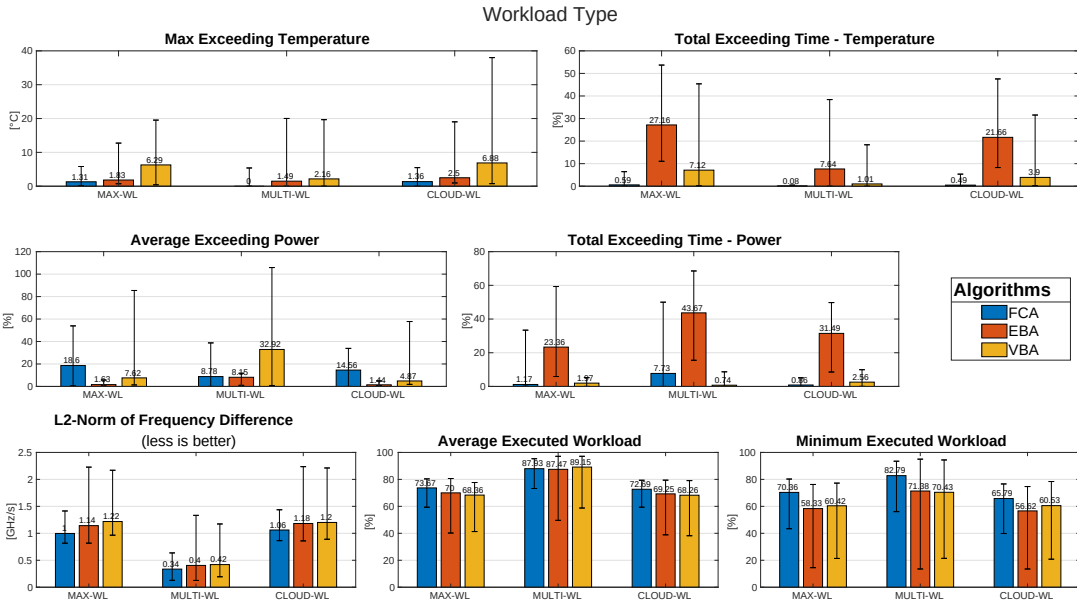


FIGURE 5.3: Control performance metrics for each workload type: vector (*MAX-WL*), mixed (*MULTI-WL*), and randomly varying (*CLOUD-WL*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

back to the use of fixed PID coefficients, which may limit adaptability to widely changing thermal conditions.

Power Regulation Metrics

In the power regulation metrics, FCA exhibits a relatively high average PW-AV (13.98%) but maintains a low PW-CT (3.25% on average). This suggests that constraint violations in power capping primarily occur during target or application phase transitions. The distribution of the PW-AV results shows a greater variability than the other two algorithms, with the PW-CT distribution containing a significant number of outliers, particularly from the MULTI-WL tests, as shown in Figure 5.3. Interestingly, performance deteriorates as the number of domains increases, except for AD, suggesting that the issue may stem from managing oscillations in coupled systems.

VBA has a similar behavior as FCA, recording an average PW-AV of 15.14% and PW-CT of 1.76%. Although the PW-CT result is the most consistent among the three algorithms, the PW-AV distribution contains numerous outliers, reaching up to 105.84% of the power budget. These outliers predominantly originate from the MULTI-WL tests. The consistency in exceeding time suggests that VBA handles gradual power transitions well, yet it suffers in scenarios with fluctuating power demands, such as the MULTI-WL tests. The relationship between the high PW-AV and consistent PW-CT values indicates that while VBA prevents prolonged periods of overconsumption, its difficulty in managing power spikes creates a risk of significant power exceedance, which can strain the power delivery system.

In contrast, EBA demonstrates the opposite behavior to the other two algorithms, achieving the most consistent and lowest average PW-AV (3.74%), but with the highest PW-CT (32.84% on average) and high PW-CT variability.

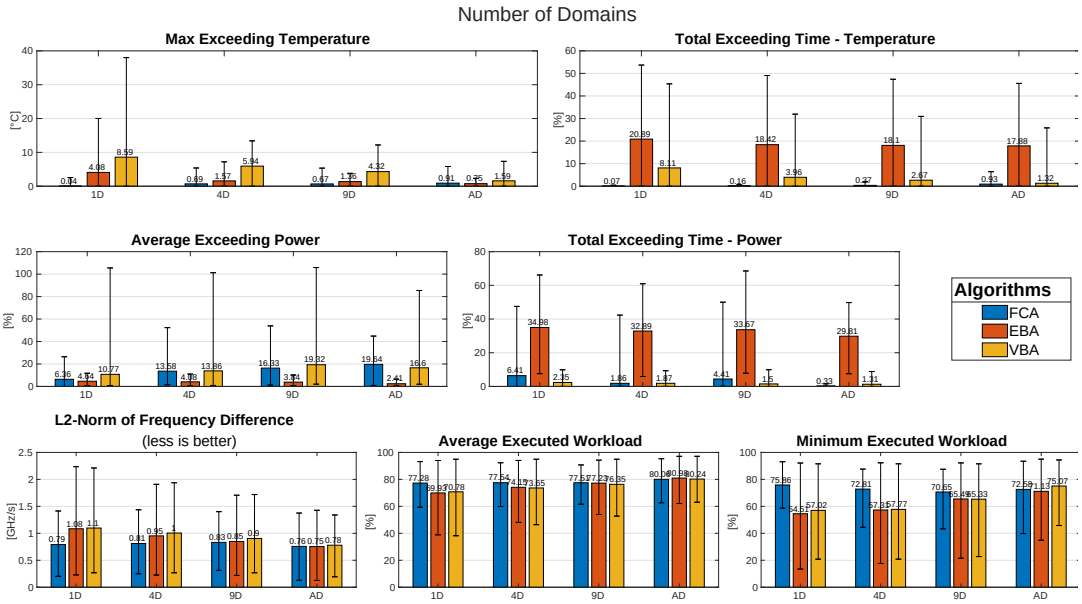


FIGURE 5.4: Control performance metrics for different domain configurations: one (*1D*), four (*4D*), nine (*9D*), and one per PE (*AD*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

Target Compliance Metrics

On average, FCA demonstrates the best target compliance (TF-CFD) performance across all tests, with the exception of the all-domains (A-D) configuration, where EBA marginally outperforms it by $0.01 GHz/s$. Specifically, FCA shows a 12.26% lower average TF-CFD value than EBA and a 15.67% lower average value compared to VBA. Both EBA and VBA improve their target compliance as the number of voltage domains increases, eventually matching FCA's performance in the A-D configuration. This result confirms EBA's strength in scenarios without voltage coupling constraints. VBA, on the other hand, tends to homogenize frequencies across PEs, which leads to favoring less power-intensive PEs (such as those running the mixed integer and floating point workloads at $2.7 GHz$). In contrast, FCA and EBA prioritize PEs executing more demanding workloads, which consume more power and generate more heat under the same conditions [20, 136].

Regarding results distribution, Figure 5.2 highlights that there exists a bimodal trend with two peaks, the lower one specifically referring to the MULTI-WL test while the other one to the other two workload scenarios. This is because, with most PEs being constantly idle, control algorithms have more available power to allocate to other PEs. Despite the bimodal considerations, FCA exhibits the smallest deviation, with the other algorithms recording results above $1.5 GHz/s$. Figures 5.4, 5.5, and 5.6 further illustrates that FCA is the most consistent in terms of target compliance as most of the variation observed is related to different workload scenarios. This consistency is confirmed by Figure 5.3, where FCA's error bars are significantly smaller than those of the other algorithms.

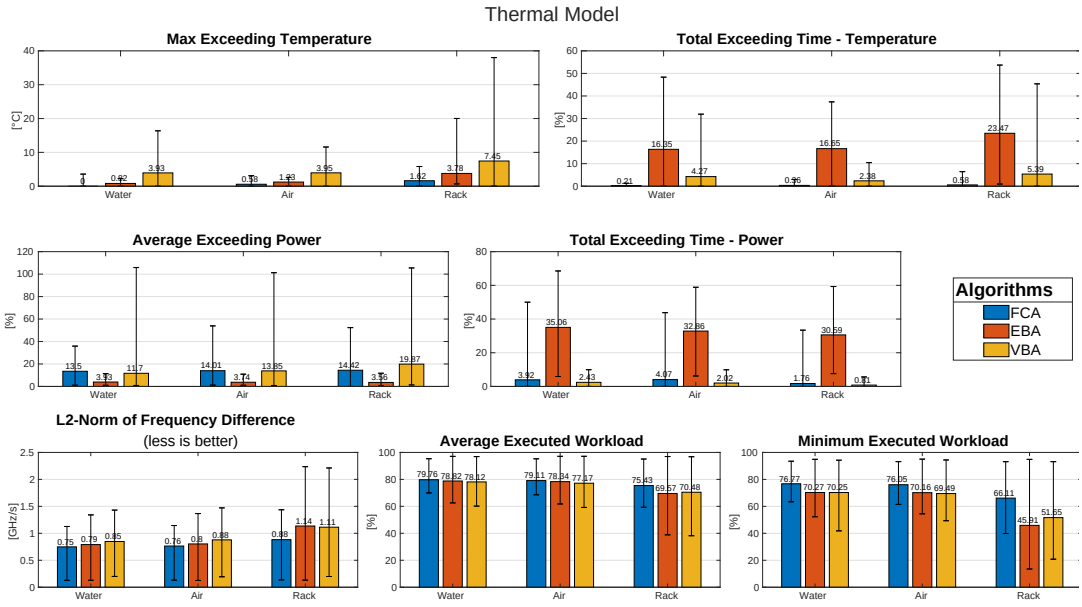


FIGURE 5.5: Control performance metrics for different thermal models: water cooling (*WATER*), air cooling (*AIR*), and horizontal rack cooling (*RACK*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

Application Execution Performance

FCA demonstrates superior application execution performance, with an average AP-AV that is 3.34% higher than EBA and 3.77% higher than VBA. Additionally, FCA achieves a 17.50% improvement in the average AP-MIN over EBA and a 14.39% improvement over VBA. Similar to the trend in target compliance, both EBA and VBA shows better results as the voltage domains' size reduces, eventually matching FCA's performance in the A-D configuration. In the MULTI-WL workload test, VBA achieves a higher AP-AV but underperforms in AP-MIN. This is attributed to VBA's tendency to homogenize frequencies across PEs, in contrast to FCA and EBA demanding workloads prioritization. This meant that the Frequency of the more demanding PEs was reduced more than the one of other PEs.

Regarding result variability, FCA exhibits smaller deviations in both AP-AV and AP-MIN results, indicating that applications running with this algorithm are less likely to experience significant performance degradation under specific conditions. The reduced variation suggests that FCA provides a more consistent execution environment, offering greater reliability.

Summary and Conclusions

The average results and their standard deviations, accounting for consistency across tests, are summarized in table 5.1. FCA achieves the best target compliance and application progress while outperforming the other algorithms in thermal regulation, with comparable power capping performance to VBA. This indicates that FCA not only more effectively applies the HLC targets but also improves application execution speed by approximately 3% while operating with tighter thermal margins. Additionally, FCA provides greater consistency in execution progress across PEs, with the least difference between AP-MIN and AP-AV compared to the other algorithms. FCA also

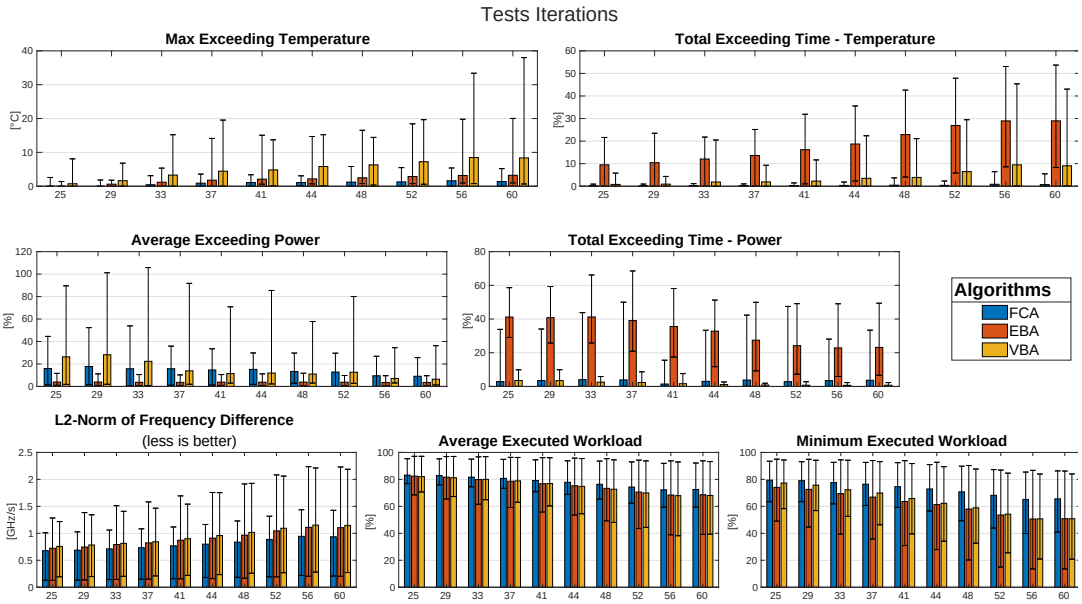


FIGURE 5.6: Control performance metrics across test iterations with relative error bars. The x-axis indicates the average initial system temperatures. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

demonstrates the most consistent performance across varying cooling configurations, domain numbers, and initial conditions, as clearly reflected in the smaller error bars shown in Figure 5.3.

In contrast, EBA's TH-CT and PW-CT distributions exhibit significant variability, indicating inconsistent behavior when non-idealities are introduced into the model. Both EBA and VBA encounter notable challenges in managing voltage coupling, particularly with larger domain configurations. Conversely, FCA's iterative root-finding approach shows superior precision in reaching the desired operating points under these types of control signal constraints.

Power regulation results further suggest that effectively capping power remains challenging, with a clear trade-off between time and value metric. Each algorithm displayed opposite trends in these metrics, highlighting the difficulty in balancing power control across different scenarios.

Conversely, the effectiveness of the proposed fuzzy-inspired thermal regulation over traditional state-of-the-art PID controllers is undeniable, as it consistently maintains superior performance across all configurations and scenarios. This demonstrates the robustness and adaptability of the fuzzy-based approach, making it a highly reliable solution for managing thermal regulation in dynamic environments.

Alg			FCA	EBA	VBA
Temperature	**TH-MAX** [$^{\circ}C$]	**Max**	5.83	20.02	38.01
		Av	0.58	1.94	5.11
		SD	2.83	3.36	5.28
	TH-CT [%]	**Av**	0.38	18.82	4.01
		SD	0.73	12.10	7.09
	Average Temp [$^{\circ}C$]		67.87	71.96	69.23
Power	**PW-AV** [%]	**Av**	13.98	3.74	15.14
		SD	10.65	3.45	25.16
	PW-CT [%]	**Av**	3.25	32.84	1.76
		SD	8.22	13.27	2.11
Execution	**TF-CFD** [GHz/s]	**Av**	0.80	0.91	0.95
		SD	0.35	0.45	0.45
	AP-AV [%]	**Av**	78.10	75.57	75.26
		SD	8.55	12.01	12.58
	AP-MIN [%]	**Av**	72.98	62.11	63.80
		SD	10.86	18.68	17.16

TABLE 5.1: Summary of the average results of the battery of tests. Standard deviation is included to show the constancy across different tests. In green the best result, in red the worst.

5.3 MPC Controllers Comparison

This secondary comparison evaluates the performance of the MPC algorithm presented in section 4.4 under similar conditions to those used in the previous comparison, offering a focused analysis of the algorithm's competitiveness and the specific innovations introduced in this work.

In the previous analysis, FCA emerged as the best-performing algorithm, positioning it as the baseline reference for this secondary assessment. It is compared against the MPC controller presented in this work, referred to as Linear Enhanced MPC (LE-MPC). Regarding a state-of-the-art MPC comparison, as highlighted in section 3.1, MPC controllers are predominantly employed as thermal regulators at the HLC level, sometimes in conjunction with other power allocation algorithms, and often not considering voltage coupling constraints. For a meaningful comparison, an MPC controller was considered with the following design:

- power consumption P_T as the control input, according to the most common configuration in the state-of-the-art
- same constraint formulation specified in (4.27), to be able to include all the LLC control problem requirements
- target P_{est} calculated similarly to EBA, including moving average voltage selection and adaptive power adjustments. This is to address the voltage coupling constraints.

To prevent any interference with the results, only this MPC algorithm will be provided with the complete array of temperature measurements, eliminating the need for implementing an observer. In the test, this MPC configuration will be referred to as Classic Thermal-Power MPC (TP-MPC).

For additional insight, a further algorithm, Linear Baseline MPC (LB-MPC), is included in the comparison. LB-MPC represents a simplified version of LE-MPC without the theta adjustments and hybridization enhancements. This inclusion serves two purposes: first, to evaluate the robustness of the linearization approach in comparison to TP-MPC, and second, as a baseline reference to assess the impact of the proposed improvements in LE-MPC.

Notably, the performance of the MPC is strongly linked to the optimization matrices Q, R, R_t, and $\mathcal{H}$. This dependency already anticipates that results may exhibit high variability; however, it also indicates the potential for achieving improved results through an iterative process to refine and learn the optimal matrices.

In this work, the following penalization parameters are selected and applied along the main diagonal of the respective matrices:

- **TP-MPC:** $q = 8$, $r = 0$, $r_t = 20$, $t_s = 1 \times 10^{-3}$, $N_{hzn} = 4$
- **LB-MPC:** $q = 18$, $r = 0$, $r_t = 20$, $t_s = 1 \times 10^{-3}$, $N_{hzn} = 4$
- **LE-MPC:** $q = 10$, $r = 7.5$, $r_t = 20$, $t_s = 1 \times 10^{-3}$, $N_{hzn} = 4$. For the matrix $\mathcal{H}$, the coefficient $h = 7.5$ is chosen, with the k_{ij} coefficients computed such that they are uniform within each domain and satisfy $\sum k_{ij} = h$. Consequently $k_{ij} = \frac{h}{n_{\mathcal{D}_j}}$

As in the previous section, results are organized by workload type (Figure 5.8), domain number (Figure 5.9), model type (Figure 5.10), and test iteration (Figure 5.11), along with the violin plot (Figure 5.7) illustrating the distribution across all test cases.

5.3.1 Results

Thermal Regulation Metrics

The TH-MAX results of both Linearized MPCs (lin-MPCs) algorithms are comparable to that of FCA, though with a wider distribution and a higher average value. TP-MPC demonstrates the best performance, exceeding the target in only a few tests with marginal values.

Conversely, the TH-CT metric reveals poor results for both lin-MPCs, with target exceedance observed in some tests for up to 40% of the test duration, with an average value of 10% and high variance. In contrast, TP-MPC and FCA show considerably better performance, with TH-CT averages of 0.15% and 6.4×10^{-5}%, respectively. This disparity may be attributed to the linearization approximation, where the time-varying offset does not fully mitigate model errors, particularly under high-temperature and high-voltage conditions. Additionally, the state observer, which supplies temperature measurements for the lin-MPCs, introduces further deviations, contributing to the inconsistencies in threshold adherence.

Different thermal models do not impact the thermal performance of lin-MPCs algorithms, provided each model was accurately 'identified' before testing. Conversely, different initial conditions, particularly with elevated starting temperatures, appear to affect the results negatively. This may be due to the previously mentioned linearization errors and observer-related deviations.

Despite the high TH-CT values for the lin-MPCs, the average exceeded temperature remain low, at $0.09°C$ and $0.10°C$ for the baseline and enhanced versions,

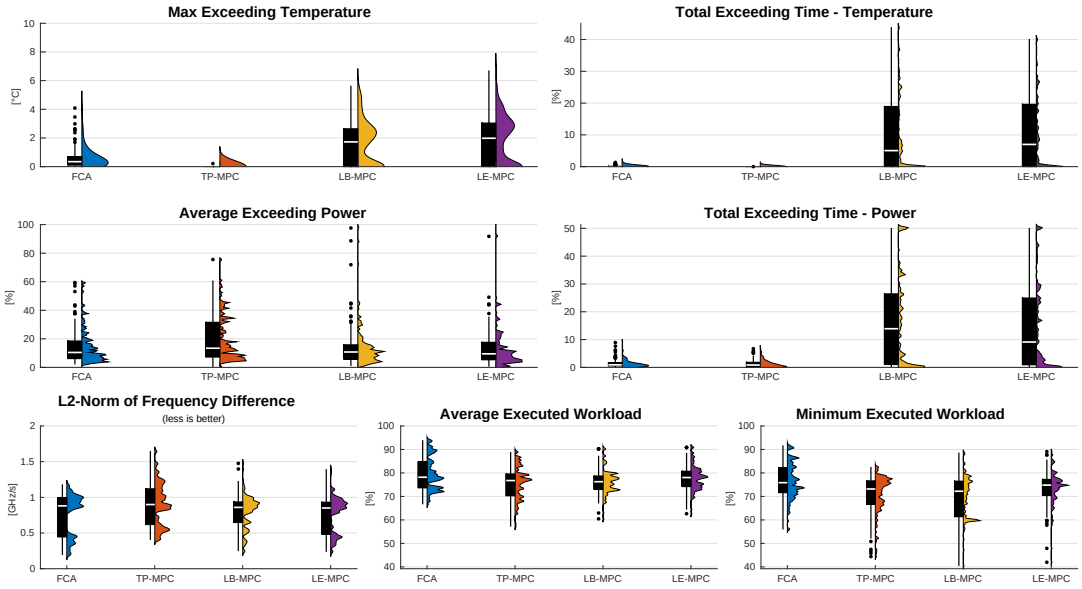


FIGURE 5.7: Overall distribution of control performance metrics for each algorithm. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

respectively. This suggests that although these factors influence TH-CT, the deviations from the thermal limit (T_L) are minimal and do not adversely impact the thermal requirements.

Overall, the performance of all four algorithms is adequate and within acceptable limits, demonstrating that the predictive capability of MPC can effectively prevent large thermal overshoots and maintain a sufficiently fast response time, even with a doubled control iteration time t_s compared to FCA. This assessment is reinforced by the consistency of results across all MPC algorithms illustrated in Figure 5.8 between the MAX-WL and CLOUD-WL tests. Workload variations are absent in MAX-WL but are significant in CLOUD-WL; thus, obtaining similar results indicates that fast control response effectively manages workload uncertainty.

Power Regulation Metrics

A similar analysis to that of thermal regulation can be applied to power regulation metrics. TP-MPC performs comparably to FCA, while both lin-MPCs yield PW-AV results similar to FCA but with a more widely spread deviation and noticeably worse results in PW-CT. Weaker lin-MPC results were anticipated due to the linear approximation. However, the addition of the time-varying offset helps keep the PW-AV values in alignment with the other algorithms analyzed.

Interestingly, referring to Figure 5.9 the worst PW-AV results are observed in the A-D configuration, whereas in all other configurations, lin-MPCs achieve PW-AV results that are comparable to or better than the other algorithms. This suggests that voltage coupling may help stabilize and mitigate control signal evolutions that might otherwise attempt to track the given target too aggressively through R_t. This pattern cannot be attributed to the $\mathcal{H}$ matrix, which is zero in the A-D configuration; additionally, both LE-MPC and LB-MPC have the same results pattern with LB-MPC $\mathcal{H}$ matrix being zero across all tests. Conversely, the opposite trend is seen in PW-CT, where larger domains tend to yield poorer results. This outcome aligns with

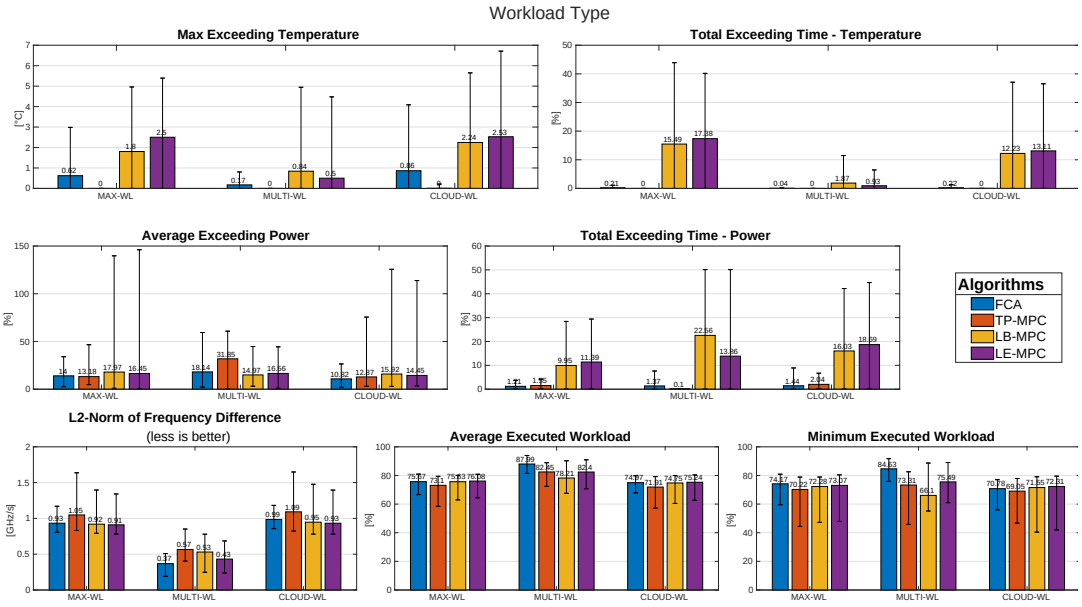


FIGURE 5.8: Control performance metrics for each workload type: vector (*MAX-WL*), mixed (*MULTI-WL*), and randomly varying (*CLOUD-WL*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

expectations for both algorithms, given the more stringent coupling constraints in larger domains.

It is also noteworthy that, similar to the thermal regulation results, power metrics remain consistent between the MAX-WL test, with its constant and uniform workload, and the more variable CLOUD-WL test. This consistency suggests that the disparity in results between FCA/TP-MPC and lin-MPCs may be due more to the response to step-input changes (such as adjustments in power budget) than to any limitation in managing rapid workload variations.

Target Compliance Metrics

The first distinction in algorithms' performance becomes evident with TF-CFD. TP-MPC performs notably worse than the other algorithms, with results that are slightly better but comparable to those obtained by EBA and VBA in the previous comparison of section 5.2. Additionally, distinctions between the two lin-MPCs start to emerge: LE-MPC achieves a 0.51%improvement over FCA, while LB-MPC has a 4.88% worse TF-CFD than FCA.

Examining specific configurations in more detail, LE-MPC shows its greatest advantages over LB-MPC in the MULTI-WL scenario and in configurations with larger voltage domains, as illustrated in Figures 5.8 and 5.9. These improvements reflect the impact of the θ-based adjustments (4.29) enabled by using P_M as control input, along with the benefits of the hybrid integration described in section 4.4.4. Specifically, the TF-CFD improvement for LE-MPC varies from 7.02% in the 1-D configuration to 5,32% in the 9-D configuration, reaching nearly identical values in the A-D configuration. Notably, this comparison test included only 16 cores, resulting in a 9-D configuration with a maximum of two cores per domain. This outcome suggests that the observed improvements stem from the inclusion of the $\mathcal{H}$

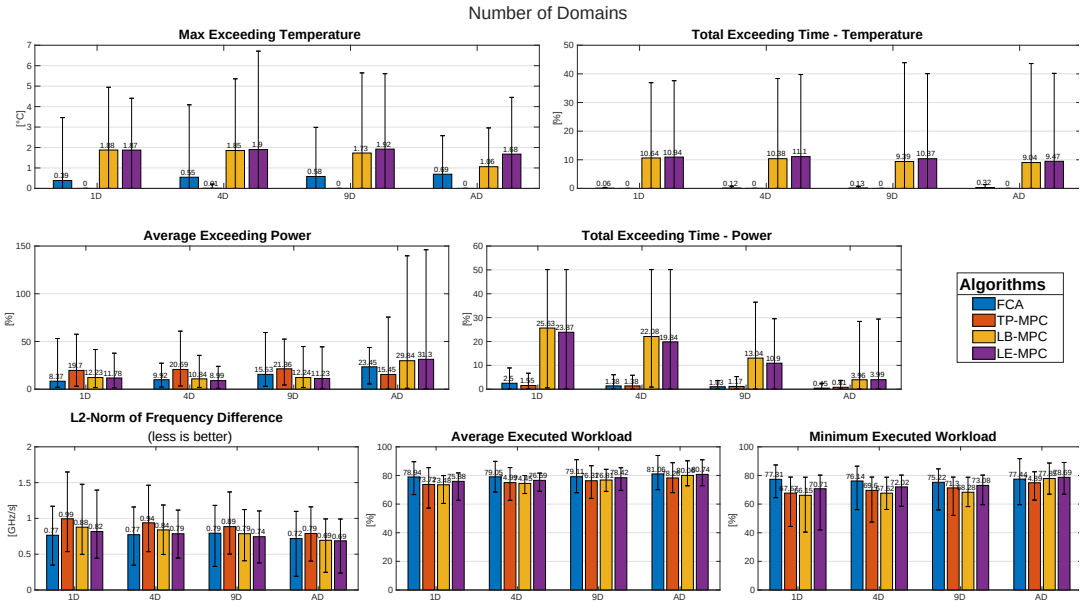


FIGURE 5.9: Control performance metrics for different domain configurations: one (*1D*), four (*4D*), nine (*9D*), and one per PE (*AD*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

matrix, which enables LE-MPC's optimization to better account for voltage coupling constraints, rather than from severe limitations in voltage selection.

In terms of workload scenarios, as shown in Figure 5.8, TF-CFD improvements between LE-MPC and LB-MPC in the MAX-WL test, where all PEs execute a constant workload, are limited to 1.04%, likely due to the gains linked to domain configurations. These improvements, however, increase to 1.56% in the CLOUD-WL scenario, where workloads vary significantly, and reach 18.61% in the MULTI-WL scenario. The primary cause of this large improvement difference may be attributed to the fact that in MULTI-WL workloads remain constant over time, allowing LE-MPC to better predict and allocate power, whereas the high workload variability in CLOUD-WL reduces the opportunities for optimized power allocation. In Figure 5.11, the TF-CFD improvement between LE-MPC and LB-MPC also appears to grow with rising initial temperatures, likely due to the more stringent thermal conditions (i.e., reduced thermal headroom) that emphasize LE-MPC's ability to select a better operating point.

TP-MPC performs worse than all other algorithms, with TF-CFD performance diminishing as initial temperatures increase and with larger domain sizes, suggesting a simpler algorithm that lacks advanced optimization features.

Comparing FCA's TF-CFD results with both lin-MPCs algorithms, FCA performs significantly better in the MULTI-WL scenario but records lower values in other workloads, with the largest difference observed in CLOUD-WL. FCA also achieves better results with larger voltage domains, but falls behind in the 9-D and A-D configurations. This trend suggests that FCA has a stronger capability for power allocation in scenarios with coupling constraints and constant workloads, while its responsiveness to fast-varying workloads appears slightly weaker. It is also noteworthy that, despite exhibiting a higher standard deviation due to the bimodal distribution of results, FCA demonstrates less overall TF-CFD variability, as shown in Figure 5.7. This indicates that FCA offers a more consistent option with comparable

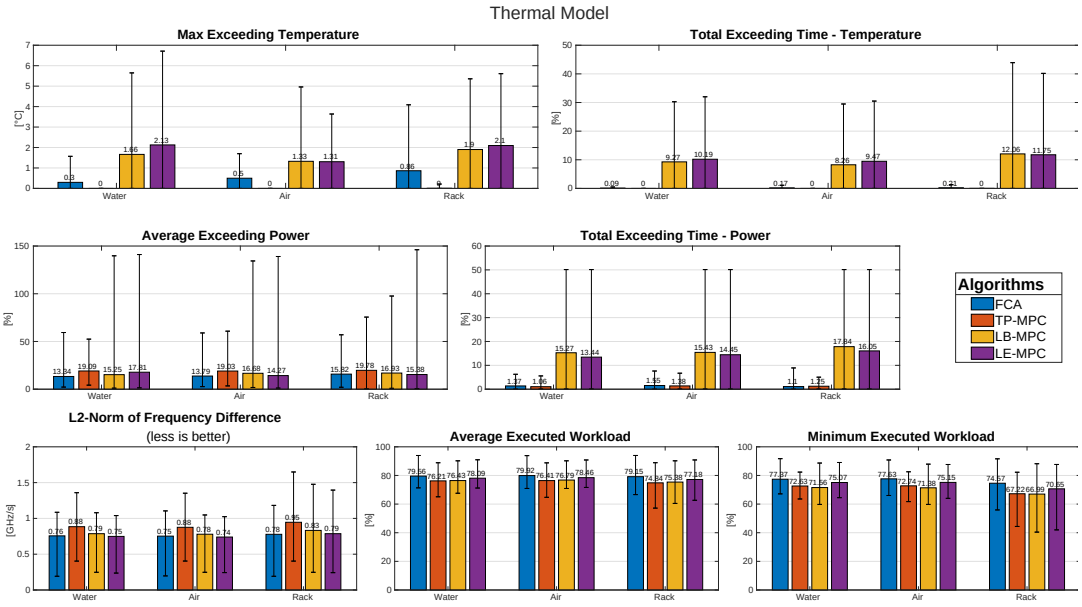


FIGURE 5.10: Control performance metrics for different thermal models: water cooling (*WATER*), air cooling (*AIR*), and horizontal rack cooling (*RACK*), with relative error bars. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

average outcomes.

Application Execution Performance

The AP-AV metrics exhibit a pattern similar to TF-CFD, supporting a comparable analysis. Nevertheless, additional insights can be gained by examining the AP-MIN metric. LE-MPC achieves more consistent results than the other algorithms, including FCA, with only a few low-value outliers. These outliers appear to stem from the stricter conditions, specifically the RACK tests with the highest initial temperature and 1-D configuration, regardless of the workload scenario.

Summary and Conclusions

The average results and their standard deviations, accounting for consistency across tests, are summarized in table 5.2.

The analysis demonstrates the robust performance of FCA, which, despite being a simpler and less computationally intensive algorithm, achieves results comparable to the more complex MPC-based approaches. FCA maintains consistent thermal, power, and performance requirements, specifically in scenarios with complex coupling constraints. This stability and efficiency make FCA a reliable control choice, even when benchmarked against the computationally demanding MPC algorithms.

The lin-MPC algorithms, in contrast, show more variation in their metrics due to their cost function J depending on fixed optimization matrices, which may limit their adaptability to diverse conditions. Although lin-MPCs exhibit higher variance in the distribution of results particularly in TH-CT and PW-CT, these stem from the limitations of the linear approximation rather than fundamental issues in meeting thermal or power requirements. TP-MPC demonstrates the highest performance in

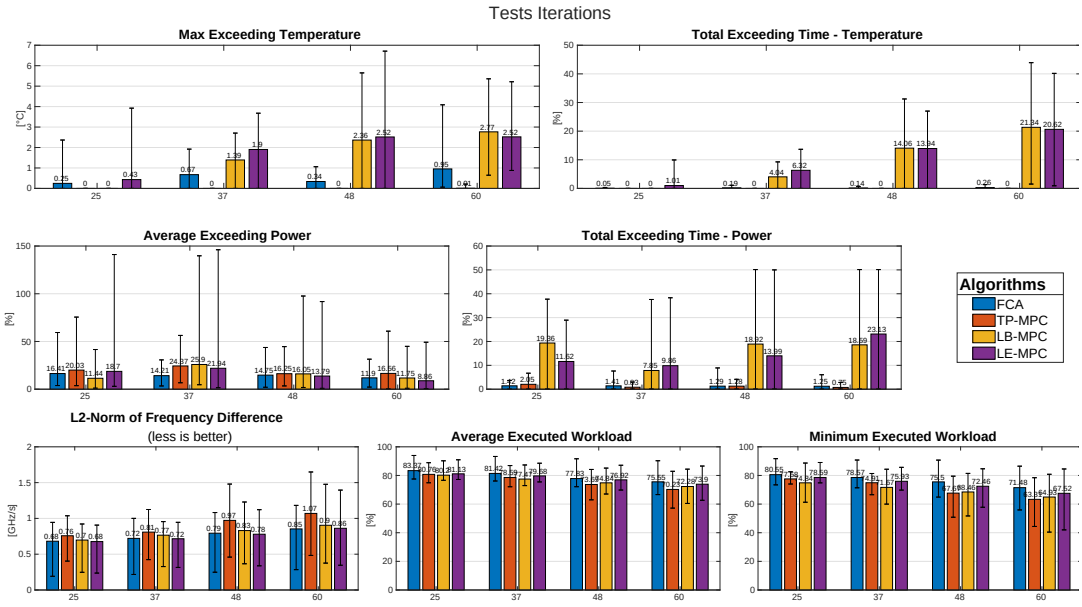


FIGURE 5.11: Control performance metrics across test iterations with relative error bars. The x-axis indicates the average initial system temperatures. The first row shows thermal metrics, the second row power metrics, and the third row target compliance and execution progress metrics.

power and thermal regulation; however, these gains are less significant when considering the consistent and acceptable results produced by the other three algorithms, coupled with TP-MPC's lowest performance metrics.

The lin-MPC algorithms exhibit weaker thermal and power regulation results compared to TP-MPC, which serves as the baseline, indicating that the linearization introduces a non-negligible degree of approximation. Future improvements could address this limitation by incorporating additional features, such as those based on robust MPC theory. Nonetheless, both lin-MPC algorithms achieve superior target compliance and application execution performance compared to the state-of-the-art reference TP-MPC, with the enhancements described in section 4.4.1 and section 4.4.4 increasing this improvement even further.

Overall, these comparisons underscore the effectiveness of FCA as a practical and efficient control solution in complex scenarios.

Alg			FCA	TP-MPC	LB-MPC	LE-MPC
Temperature	TH-MAX [°C]	Max	4.09	0.21	5.65	6.71
		Av	0.55	$1.5 \cdot 10^{-3}$	1.63	1.84
		SD	0.70	0.012	1.54	1.65
	TH-CT [%]	Av	0.16	$6.4 \cdot 10^{-5}$	9.86	10.47
		SD	0.22	$7.7 \cdot 10^{-4}$	11.79	11.65
	Average Temp [°C]		66.28	64.33	66.43	66.50
Power	PW-AV [%]	Av	14.32	19.30	16.29	15.82
		SD	11.76	15.23	22.15	23.53
	PW-CT [%]	Av	1.34	1.23	16.18	14.65
		SD	1.46	1.37	15.81	15.40
Execution	TF-CFD [GHz/s]	Av	0.76	0.90	0.80	0.76
		SD	0.29	0.29	0.23	0.25
	AP-AV [%]	Av	79.54	75.82	76.20	77.91
		SD	6.96	6.72	4.94	5.04
	Ap-MIN [%]	Av	76.53	70.86	69.98	73.62
		SD	7.41	7.93	8.66	6.88

TABLE 5.2: Summary of the average results of the battery of tests. The standard deviation indicates the consistency of each algorithm across different tests. The best results are in green, the worst in red.

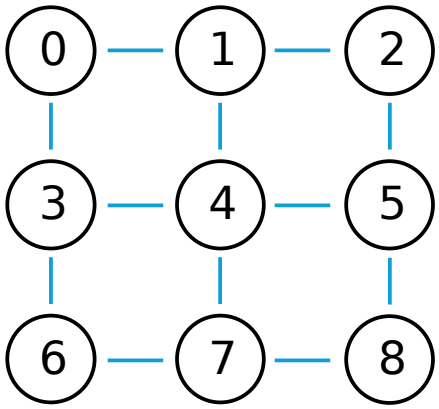
5.4 Distributed Control Analysis

The application of distributed control strategies to HPC systems allows to solve the problem of allocating a power budget across multiple interconnected controller nodes. Leveraging ROS 2 [93], an open-source platform optimized for modular and real-time communication, this work analyzes the efficiency and responsiveness of power distribution across a network of controllers within a distributed control architecture.

This test considers a network configuration comprising nine controllers, each represented as a node in a square grid topology with direct horizontal and vertical communication links to adjacent nodes (see Figure 5.12). Each node is assigned a constant power target and workload information, both of which remain fixed throughout the test. The optimization objective is defined by the function in eq. (4.47), and the gradient tracking algorithm described in section 4.5.2 is used to coordinate the allocation of power among the nodes.

The enforced network-wide power budget constraint is set to approximately 75% of the sum of the power targets of all nodes. A step size of $\gamma = 0.1$ is applied.

The results in Figure 5.14b demonstrate that this network configuration successfully satisfies the shared power budget constraint within 30 controller iterations. It is worth noting that, for each node i the initial values of the P_j elements in x_i relative to the other nodes are initialized to a minimum value of $0.5W$, significantly lower than their actual target values. Figure 5.13 further illustrates this behavior, where the thinner lines converge to the converged P_i values of each node within approximately 30 iterations.



(A)

0.6	0.2	0	0.2	0	0	0	0	0
0.2	0.44	0.2	0	0.16	0	0	0	0
0	0.2	0.6	0	0	0.2	0	0	0
0.2	0	0	0.44	0.16	0	0.2	0	0
0	0.16	0	0.16	0.36	0.16	0	0.16	0
0	0	0.2	0	0.16	0.44	0	0	0.2
0	0	0	0.2	0	0	0.6	0.2	0
0	0	0	0	0.16	0	0.2	0.44	0.2
0	0	0	0	0	0.2	0	0.2	0.6

(B)

FIGURE 5.12: Figure 5.12a on the left illustrates the 9-node square network configuration, where each line represents a duplex communication. Figure 5.12b on the right shows the adjacency matrix used in the test.

Upon reaching consensus, the P_i values continue to evolve as the gradient tracking algorithm minimizes the sum of the nodes' cost functions while preserving the power budget constraint. This evolution is reflected in Figure 5.14a, where the y_i values, which store the gradients of the cost functions, decrease over time as the optimization progresses.

Additionally, Figure 5.13 highlights that nodes with higher power targets (i.e., those handling more demanding workloads) experience less reduction in their P_i^* targets compared to those with lower power demands, according to the r_i parameters in eq. (4.47).

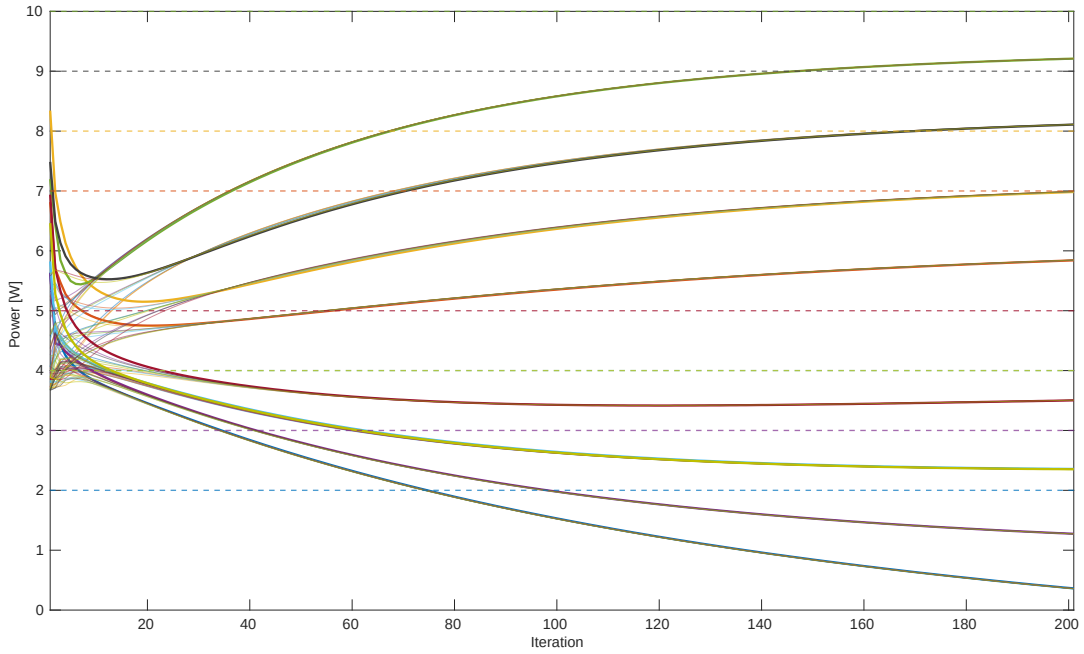
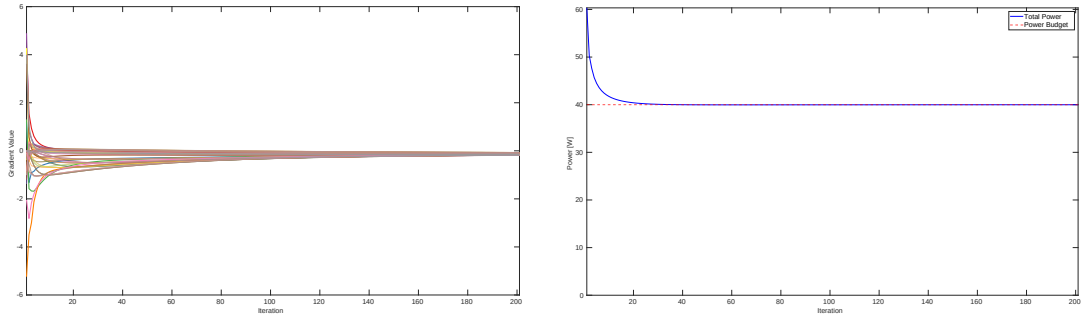


FIGURE 5.13: Evolution of the power vector x_i of each node under the gradient tracking algorithm. Lines of the same colors correspond to a single node. The thicker lines represent the node-specific P_i values, while the thinner lines indicate P_j values of other nodes. Dashed lines denote the power targets P_i^*.



(A) (B)

FIGURE 5.14: Figure 5.14a (left) shows the y_i gradient tracking value of the algorithm. Lines of the same colors correspond to a single node, with thicker lines representing the node-specific y_i elements and thinner lines indicating the y_i elements of other nodes. Figure 5.14b (right) illustrates the total network power consumption (blue line) converging to the power budget threshold P_B (red dashed line).

Chapter 6

Hardware in the Loop (HIL) Implementation

In Chapter 5, the comparison between algorithms was conducted in a Model in the Loop (MIL) environment. MIL is a method for testing the control action within a simulated software environment, allowing developers to validate the performance and correctness of their algorithm before deploying it on actual hardware systems. This is only the first step in development as it may not fully capture real-world hardware interactions, which can lead to differences in performance during actual deployment [119].

The next step is to transition from simulation to hardware testing. This shift allows for a more comprehensive evaluation of the control algorithms by assessing their performance in a more realistic scenario. By conducting tests directly on the hardware, it is possible to validate the system's response to various inputs and ensure that real-time constraints are met, while other tasks, such as communication and data transfer, are carried out simultaneously. To perform this level of testing, a Hardware in the Loop (HIL) framework was set up [100].

HIL testing adoption has been rising as an important tool in the development and validation of control systems across various industries. By embedding physical hardware components into a real-time simulation environment, HIL offers an alternative way to test and fine-tune the overall system behavior under a wide array of testing conditions. This technique bridges the gap between pure simulation and real-world application, allowing developers to validate control algorithms, hardware functionality, and system integration without exposing the actual system to potentially damaging scenarios.

The HIL approach enables rapid prototyping and iterative testing, removing the need for repeated physical system builds during early design stages, and it allows tests to be conducted under particular operating conditions that would be difficult or costly to recreate in real-world environments. The ability to test failure modes and critical situations in a safe virtual environment is particularly invaluable for safety-critical systems such as HPC power and thermal control, where failures would result into damaging a very expensive early testing sample [134].

In this work, a HIL framework is implemented using a Xilinx Ultrascale+ Field-Programmable Gate Array (FPGA), specifically the ZCU102 board [167], to emulate and validate real-time control in HPC systems. The ZCU102 integrates a powerful Processing System (PS) with four Arm Cortex-A53 cores and a dual-core Cortex-R5F Real-Time Processing Unit, alongside a Programmable Logic (PL) region that provides extensive reconfigurability. This setup enables simultaneous execution of the control firmware, running on the PL, while plant models are simulated on the PS, facilitating real-time interactions. The framework's setup involves programming the FPGA with the TPC hardware's bitstream, booting a lightweight version of Linux OS on the

PS, loading the control firmware into the FPGA's memory, and then launching the simulation on the PS cores. This setup establishes a Hardware-Software Co-design Framework that closely replicates real hardware conditions, enabling comprehensive testing and evaluation of control algorithms in an HPC context. It also favors iterative improvements across all components of the LLC controller subsystem, including hardware, software, and control action [149].

6.1 Hardware-Software Co-design Framework

In HPC systems, effective thermal and power management relies on a tightly integrated TPC controller (2.3). An HIL framework enables simultaneous improvements of both hardware and firmware components, while ensuring that the control algorithm keeps meeting real-time deadlines and performs as desired. This co-design approach becomes critical when implementing complex algorithms, where both hardware and software must coordinate to provide precise and fast responses to rapidly changing conditions.

An iterative hardware-software co-design process allows for continuous refinement and adaptation of the TPC to meet performance requirements. By adjusting hardware resources, such as computing units or interrupt handling subsystems, in tandem with firmware structure, scheduling policies, and control algorithms, developers can repeatedly test each component to achieve the control objectives effectively. This iterative process is particularly beneficial in early design stages, where TPC and system hardware are not yet available, leveraging FPGAs as a HIL emulation platform. This framework is valuable for developing and testing control firmware in scenarios that closely resemble real-world operating conditions, to iteratively test its design along with all other components [134].

The developed Hardware-Software Co-design HIL framework consists of three key components:

- the **TPC hardware**. Serves as the execution platform for control actions, interacting directly with simulated sensors and setting operating points for the simulated actuators.
- the **Firmware**. Manage the TPC's core functionalities, implementing the control algorithm to execute real-time adjustments based on sensor data and control requirements.
- the **Simulation**. Simulates the controlled system (or "plant"), allowing for rapid prototyping of control architectures across diverse systems and specific edge cases, ensuring safe testing without dependence on physical hardware.

6.2 The TPC Hardware: ControlPULP

The HIL framework leverages ControlPULP [111] as the dedicated TPC hardware microcontroller, chosen for its scalable architecture and compatibility with high-performance control demands in HPC environments. ControlPULP is an open-source RISC-V-based platform designed for the role of power and thermal management controller in modern HPC processors. Based on the PULP project [131] it is tailored to meet the control requirements of increasingly complex modern chips offering a scalable, parallel architecture optimized for real-time capabilities.

The motivation behind ControlPULP choice lies in the growing computational needs for real-time, fine-grained control of power and thermal characteristics in modern processors. With the increasing number of PEs in HPC systems, traditional single-core microcontroller-based TPCs are no longer sufficient to provide the necessary computational capabilities while maintaining low-power characteristics and having minimal footprint on the die [110].

ControlPULP consists of two main subsystems: a single manager core and a multi-core programmable cluster accelerator. The manager core, based on the CV32E40P RISC-V processor, is responsible for handling fast interrupt-driven tasks, while the programmable cluster accelerates the parallel computation of real-time power management policies. The architecture includes a dedicated Direct Memory Access (DMA) engine to speed up data transfers, a specialized real-time interrupt controller, the Core-Local Interrupt Controller (CLIC), which provides low-latency interrupt handling with pre-emption support, and an advanced dedicated Floating Point Unit (FPU) for each core of the cluster.

ControlPULP's primary goal is to provide a platform capable of implementing advanced Multiple-Input Multiple-Output (MIMO) control algorithms on a large number of elements with predictable performance and low latency. To communicate with the system, it includes a robust peripheral subsystem designed to manage off-chip communication and system-level power management. This subsystem includes a specialized I/O data engine unit known as the μDMA intellectual property (IP), which enables efficient and autonomous data transfers between off-chip components and the ControlPULP's L2 SRAM. This feature minimizes the computing load on the manager core, allowing it to focus on other tasks. The peripheral subsystem supports industry-standard interfaces such as Adaptive Voltage Scaling Bus (AVSBus) and Power Management Bus (PMBus), which extend traditional I2C and SPI protocols for the digital monitoring and management of voltage and power rails. These interfaces are crucial for coordinating with VRMs, ensuring that voltage and power delivery to the processor cores are dynamically adjusted in response to changing workload demands, power budgets, and thermal conditions [110].

In addition, ControlPULP integrates multiple I2C master/slave interfaces to enable communication with the BMC and other board-level controllers. To this end, it is able to support communication through the Platform Level Data Model (PLDM) and the Management Component Transport Protocol (MCTP) transport layers.

For the on-die communication, ControlPULP offers a powerful DMA engine capable of 2D stride transfers. This allows for efficient data movement, particularly when reading data from the PVT sensors with non-contiguous address mappings, such as those found in large-scale HPC systems. By offloading data acquisition tasks to the DMA engine, ControlPULP significantly reduces the burden on the processor cores, enabling the controller to acquire sensor data and compute control decisions in parallel, enhancing both performance and responsiveness [110].

With its extensive peripheral subsystem, ControlPULP offers high integrability, positioning it as a flexible and adaptable solution for power management across large-scale systems, including HPC.

6.3 The TPC Firmware

Based on the capabilities of the ControlPULP hardware and the timing constraints outlined in section 3.3.4, the firmware is designed to meet requirements important for reliable operation in high-performance environments [147]:

- **Computationally Lightweight**: the firmware must be designed to ensure responsive execution on resource-constrained hardware, allowing it to meet real-time deadlines without introducing latency or computational bottlenecks.
- **Real-Time Capabilities**: to manage the varying timing and execution requirements of different operations—particularly control-related tasks—the firmware is designed to ensure a deterministic, reliable execution with robust recovery from potential failures. This includes maintaining prompt task scheduling with prioritized execution, responsiveness to events, and prevention of execution overruns to meet all time-sensitive demands.
- **Safety Features and Interrupt Management**: robust safety features are integrated to manage critical interrupts effectively. These include priority-based interrupt handling, protection against race conditions, and mechanisms for managing fault conditions in real-time, ensuring stable and reliable control action without risking system integrity.
- **Modularity, Scalability, and Portability**: the firmware's modular design allows each component to be independently upgraded or replaced, offering flexibility for its modification and improvement. It has to be built for scalability to manage changing system complexity or integrate additional modules as needed, and portability to further ensure compatibility across diverse hardware configurations.

Together, these characteristics form a cohesive and robust firmware foundation, aligning with the requirements of responsive and reliable control in HPC systems.

6.3.1 Real-Time

In HPC systems, real-time firmware characteristics are essential for maintaining a safe and stable operational state. Key requirements include control interval management, task scheduling with prioritized execution, pre-emption and task swapping, and efficient handling of interrupts during execution. Additionally, robust response to deadline misses is fundamental to ensuring continuous and reliable system operation. The firmware has to ensure that the control action executes with minimal latency, responding to power level variations and thermal conditions promptly. Such rapid response is crucial for mitigating risks associated with excessive heat or power, which can otherwise compromise hardware integrity and lead to unsafe conditions [98, 150].

To meet these stringent real-time requirements, this work incorporates a real-time kernel layer as the foundation of the TPC firmware architecture, with the firmware developed directly on top of this layer. The software stack, shown in Figure 6.1, leverages FreeRTOS as the chosen kernel to provide real-time capabilities such as tasks scheduling, prioritization, and interrupt management.

FreeRTOS

FreeRTOS is a Real-Time OS (RTOS) designed for embedded applications requiring minimal overhead, high reliability, and efficient task scheduling. Its lightweight kernel provides pre-emptive multitask scheduling, allowing developers to manage priorities, which is particularly beneficial for systems requiring predictable response times. The architecture of FreeRTOS supports a wide range of microcontrollers and platforms and can be readily adapted to new ones through the development of a dedicated C porting file [18]. Its open-source nature further enhances its flexibility and

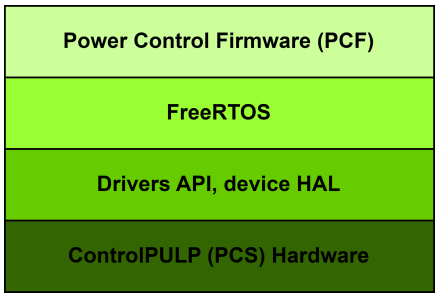


FIGURE 6.1: TPC software stack. The application control policy executes on top of FreeRTOS, which controls the hardware with target-specific drivers and HAL Application Programming Interfaces (APIs).

suitability for this work, allowing for a seamless recompilation of the kernel for the RISC-V ISAs, including ones with an extended instruction set such as ControlPULP.

Key features of FreeRTOS include support for task prioritization, inter-task communication, and synchronization, achieved through mechanisms such as queues, semaphores, and mutexes. FreeRTOS also incorporates memory management strategies designed for small-footprint systems, including static and dynamic allocation options. Additionally, the kernel includes a configurable tick rate, allowing developers to fine-tune the timing resolution according to the system's real-time requirements [19]. This combination of low overhead, determinism, and configurability positions FreeRTOS as a suitable solution for applications demanding strict timing and reliability in their control actions.

The tick rate in FreeRTOS influences the timing of an internal management function responsible for kernel operations and task scheduling, including task context switch management. The SysTick, which is the event triggering the tick change, is typically configured at a frequency of $1kHz$. To achieve specific periodic timing solely with FreeRTOS, one approach is to reduce the SysTick frequency. However, this increases the RTOS overhead, which may adversely impact the performance of the TPC [15]. Linking the control task's execution interval directly to FreeRTOS introduces some limitations: (i) the control structure becomes tied to the FreeRTOS structure, making control actions more susceptible to RTOS runtime issues; (ii) the control task's period can only be adjusted to integer multiples of the SysTick period at runtime, limiting flexibility. An advanced configuration would allow for dynamically adjusting the frequency with finer granularity—decreasing it if deadlines are not met or increasing it when sufficient slack time is available to improve performance; (iii) decreasing the SysTick frequency to lower the overhead could not be done. FreeRTOS also supports internal software timers, but these operate as multiples of the SysTick, thus incurring the same issues [19].

However, FreeRTOS enables the use of dedicated hardware timer interrupts by appropriately configuring the `FreeRTOSConfig.h` file. In the work, a hardware timer was configured to trigger an Instruction Service Routine (ISR) at a specific, runtime-adjustable frequency to achieve the desired periodicity without the limitations discussed above. To differentiate it from the SysTick, this interval is called Timer Application Periodicity (TAP). Within the TAP ISR, the desired task is signaled using the FreeRTOS API function `vTaskNotifyGiveFromISR()`. Task notifications provide the fastest and most efficient method of blocking and unblocking tasks, allowing them to interact with each other and synchronize with ISRs without requiring additional communication objects. This approach makes task notifications faster and more memory-efficient than other kernel objects that achieve similar functionality.

While task notifications have some limitations, these do not impact the requirements of this implementation.

Tasks also need to exchange data, commands, and status information. In a real-time system with safety requirements, race conditions and data corruption must be prevented by following a carefully designed sequence. FreeRTOS provides two synchronization objects for this purpose: binary semaphores and mutexes. A binary semaphore operates as a token representing the availability of a protected resource. Mutexes are binary semaphores with a priority inheritance feature, with the downside that they cannot be used within ISRs. Priority inheritance minimizes the effects of priority inversion by raising the priority of a task holding the mutex to match the highest priority of any blocked tasks attempting to acquire it. For data exchanged between tasks, mutexes are employed, with one mutex assigned per global variable accessed by multiple tasks. For variables accessed by both tasks and ISRs, binary semaphores are used instead, allowing event-driven triggers, such as signaling when data is ready or a transaction is successfully completed. When binary semaphores are used, careful attention is given to ensuring they block only a single task without nested blocking calls, thereby avoiding priority inversion [150].

The array of features and configurable options, combined with FreeRTOS's lightweight design, static memory management, fast response times, and safety mechanisms, as well as its open-source nature enabling straightforward compilation for ControlPULP hardware and future enhancements, establish FreeRTOS as an ideal choice for the TPC firmware.

Scheduling Characteristics

The control action is divided into three distinct tasks—Fast, Periodic, and Slow Control Tasks—each with specific execution intervals, priorities, and contexts. This division helps to separate computational and timing requirements, as detailed in section 3.3.4, enabling an optimized trade-off between accuracy and responsiveness for each task. High-responsiveness features are allocated to the Fast Control Task (FCT), which necessitates being computationally lightweight to avoid impacting overall system schedulability. Control features requiring more intensive computation are assigned to the Slow Control Task (SCT), which balances this demand with lower execution interval frequency, while the primary control actions for thermal and power regulation are managed within the Periodic Control Task (PCT).

This structure results in a real-time, priority-driven scheduling environment with static task priorities. To ensure the FCT meets its deadlines and maintains responsiveness, task preemption within the kernel is required.

Due to ControlPULP's architecture, which includes both a manager core and an accelerator, as described in section 6.2, either the PCT or SCT is offloaded to the accelerator based on their respective execution interval-to-computation time ratios. This allocation introduces a degree of hardware parallelism, simplifying the main core's scheduling by effectively managing only two control tasks, though it requires additional measures to maintain data coherence.

To improve schedulability within the firmware, particular attention is given to maintaining harmonic relationships between task execution time intervals. Using harmonic frequencies in real-time scheduling minimizes latency and interference by aligning task executions within predictable time slots. This approach simplifies achieving a feasible schedule, particularly in priority-based systems, and reduces the computational load associated with frequent preemptions [162]. In this work, the proposed frequency for the FCT is $10KHz$, for the PCT is $2KHz$, and for the SCT

is $200Hz$. The addition of a task that manages communications with $1KHz$ is also theorized.

6.3.2 Code Structure

Based on the requirements detailed in section 6.3, the firmware is organized into modular components designed to be independently replaced, requiring no changes to other modules and thus remaining self-contained. Each primary functionality of the firmware is encapsulated within its own module:

- **Core Module**: contains the fundamental firmware operations and basic functionalities
- **RT Module**: serves as a translation layer that encapsulates the real-time (RT) kernel, allowing for future upgrades or replacement with alternative solutions
- **Control Module**: houses the control action, ensuring it operates independently from other components, and allowing to easily test different algorithms
- **Target Module**: acts as an interface between the target hardware and drivers, and the firmware code, enabling flexible hardware changes without affecting other modules
- **System Module**: interfaces the system plant with the firmware, facilitating integration and communication (e.g. sensors and actuators)
- **Additional Libraries**: the firmware can include supplementary modules and libraries, such as a library for mathematical functions and another for communication protocols. These are customizable and less essential to core functionality.

This structure enables rapid swapping and modification of individual components without compatibility issues. The system and target modules, in particular, facilitate quick replacement of the plant and the controller hardware respectively, which is especially valuable for testing and debugging.

Each module is designed with a standardized internal structure. In addition to the classic `src/include` organization, three additional files have commonly defined rules. A `cfg_*.h` file is included within each module, containing definitions and parameters that determine the module's behavior. This configuration file influences only its specific module and allows for rapid iterative testing by adjusting these parameters without modifying the module's internal code. The asterisk in the file name is intended as the module's unique name. A `pcf_*.h` file holds the prototypes for all functions within the module that are called by external entities, serving as a stable API interface that remains unchanged when the module is replaced, upgraded, or modified. Some modules may include a `*_types.h` file that defines custom types specific to the module, ensuring consistency in data types across the firmware.

Additional consideration was given to the control module, as it contains the code likely to be the most computationally demanding, necessitating parallelization. To avoid maintaining multiple versions of the control algorithm (one for single-core and another for parallel execution on the accelerator) and to allow run-time flexibility on accelerator execution, a mechanism was developed to parallelize the iterations within the control algorithm. This mechanism only requires that functions implementing the control algorithm adhere to a standardized prototype:

```
1    float fNameOfTheFunction(int i_start, int i_end, int
        i_increment, struct ctrl_task_index* tptr, void** args);
```

The first three variables define the start, increment, and end values for any loops in the code. `tptr` is a struct variable that contains all pointers and information related to the control environment, including an indicator for parallel execution, which allows for conditional use of parallelized math library functions. The `args` parameter provides additional arguments for the function, while the `float` output is used when a final summation is required.

The following `function_param` struct provides a configuration interface for setting up the function for execution, defining its parameters and settings:

```
struct function_param {
    float (*f)(int, int, int, struct ctrl_task_index*, void
        **);
    int total_iterations;
    int parall_num;
    int exec_time;
    void** ptr_args;
    struct ctrl_task_index *tptr;
    float* return_value;
};
```

where `f` represents the function to be parallelized, `total_iterations` defines the total number of iterations in the internal loops, and `parall_num` indicates the number of processing units available to run the parallelization. `exec_time` stores the measured total execution time of `f` on a single core, `ptr_args` is the pointer to any additional arguments required by the function, and `return_value` is the pointer to the function's return accumulation value. The macro used to put this function in execution is:

```
void MACRO_CP_CLUSTER_PARALLELIZE(struct function_param*
    param) {
     int exec_gain = param->exec_time / (int)param->
        parall_num + CP_CLUSTER_OVERHEAD;
     if ( (param->parall_num>1)&&(exec_gain<param->exec_time)
         ) {
         if (param->return_value == NULL){bTargetClusterFork(
            vDummyFork, (void*)param, param->parall_num);}
         else {float accum[MAX_NUM_CLUSTER_CORE] = {0}; float
             *ptr_hold = param->return_value; param->
            return_value = (void*)accum;
             bTargetClusterFork(vDummyFork, (void*)param,
                param->parall_num);
             *ptr_hold = VD_ZERO; for(int ccore=0; ccore <
                param->parall_num; ccore++) {*ptr_hold +=
                accum[ccore];}} }
     else {vStartMeasure(); if (param->return_value == NULL)
        {param->f(0, param->total_iterations, 1, param->igl,
         param->ptr_args);}
         else{*(param->return_value) = param->f(0, param->
            total_iterations, 1, param->igl, param->ptr_args
            );}; param->exec_time = vStopMeasure();}
}
```

where `bTargetClusterFork` is the RT module function called to parallelize execution within the cluster, and `vDummyFork` manages the computation of start, end, and increment values as follows:

```
void vDummyFork(void *args)
{
```

```
        struct function_param* param = (struct function_param*)
            args;

        int i = (int)lTgtGetClusterCoresNumber();
        int s = (int)lTgtGetClusterCoreId();

        int it_chunk = param->total_iterations / i;
        if ( (param->total_iterations % i) != 0)
        it_chunk += 1;

        varFor e = s + i * it_chunk;

        // Handle odd core number
        if (e > param->total_iterations)
        e = param->total_iterations;

        if (param->return_value == NULL)
        {
            param->f(s, e, i, param->igl, param->ptr_args);

            vTargetClusterTeamBarrier();
        }
        else
        {
            param->return_value[s] = param->f(s, e, i, param->
                igl, param->ptr_args);

            vTargetClusterTeamBarrier();
        }
    }
```

Naturally, scheduling and dynamically relocating the control function at run-time introduces coherence challenges. Information from one control iteration must be preserved for use in subsequent iterations. While the ControlPULP cluster includes DMA capabilities to facilitate transfers, these operations must be programmed in advance to manage data movement effectively.

This adaptable code structure establishes a foundation to enable updates and iterative test capability needed to efficiently use the Hardware-Software Co-design framework. The modules' self-contained design supports ensures that even complex changes can be managed with minimal impact on the overall firmware architecture.

6.3.3 Safety Requirements

Safety is a fundamental aspect of firmware design, particularly in real-time control systems, where unexpected behavior or data inconsistencies can compromise system stability. Essential safety features include mechanisms for detecting, handling, and recovering from internal execution errors, as well as logging and signaling any issues. In the event of critical failures, the firmware should incorporate fail-safe modes to safely transition the system to a power-down sequence [147].

Watchdog timers and designated execution checkpoints are implemented to continuously monitor the firmware's operational state, detecting potential deadlocks or hangs in task execution [120]. Additionally, memory integrity checks further contribute to safe operation by preventing corruption of data and instructions.

In the context of HPC DPTM firmware development, special attention is given to validating operational results, as well as verifying the feasibility of incoming

measurements and outgoing control commands. Where supported by TPC hardware, privileged operations are also employed during communication with the HPC system to further prevent information gathering or data tampering.

Currently, most of these safety features are not yet implemented, as the primary focus has been on developing and analyzing the control algorithms. Nonetheless, their implementation is essential in further development of the firmware toward a finalized and reliable product.

6.4 The HPC Chip Simulation

The Simulation serves as a model of the plant (an HPC chip in this case), replicating its thermal and power dynamics, sensor readings, and actuator effects on the system, as well as target variations and workload traces. This simulation allows for testing control algorithms in a safe, adjustable environment, enabling developers to refine control strategies and evaluate performance across different scenarios without the need for physical hardware.

The plant model is adapted from the one described in Chapter 2 and converted into an executable file that can operate on a low-computation system, like the PS cores of the FPGA. C was selected as the coding language for the simulation for its performance efficiency, general applicability, and due to familiarity with the language, although alternatives like C++ or Rust could also have been viable choices [118]. The PS of the FPGA runs a lightweight custom Linux OS distribution, allowing the simulation to execute as a privileged, compiled application [110].

The simulation is divided into multiple threads to enhance the computational capacity through parallelized execution. The threads' composition is as follows:

- **Thermal Model Thread**: the most computationally demanding thread, responsible for running the thermal model
- **Power and Performance Thread**: manages the performance model through the given operating frequency and compute the power model
- **Data-Saving Thread**: records data throughout the simulation for later analysis of results, reading them directly from the DRAM
- **Simulation Manager**: coordinate the simulation while providing time-varying HLC targets based on a predefined trace.

Linux timers manage the real-time aspects of the simulation, waking each thread at specified intervals, while a custom implementation of pthread mutexes is used as binary semaphores for thread synchronization. Data exchanges with the TPC occur through shared DRAM addresses to ensure simplicity and rapid communication. This setup requires specific Linux implementations with root privileges and appropriate kernel settings.

The configuration of the simulated system is provided via a JSON file, while the trace inputs for HLC target points and workload are provided as CSV files. The data-saving thread, after gathering data directly from the DRAM at specific intervals, transmits it to a database using the MQTT communication protocol. Custom publisher and collector components, based on the Mosquitto library, were developed to handle high data volumes and throughput efficiently. Data storage and visualization are managed through Examon.

Examon is an open-source framework used for real-time monitoring and analysis of performance and energy metrics in HPC systems [34]. It enables the collection

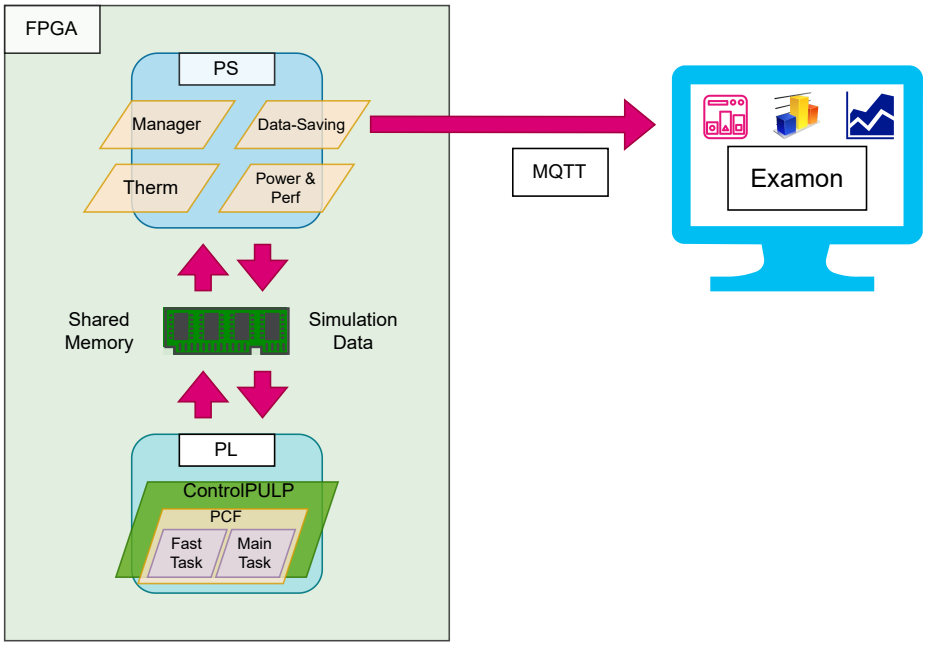


FIGURE 6.2: Representation of the HIL Co-design Framework. The chip simulation runs on the PS cores through 4 threads, while on the PL ControlPULP is instantiated. On ControlPULP, the real TPC Firmware (PCF) runs with the control algorithm split into the Fast and Main Tasks. Simulation Data are exchanged through shared memory, and they are collected and sent to an external node by the Data-Saving thread.

of a large volume of data from various sources, such as hardware sensors and software counters, and organizes it within a time-series database. Utilizing lightweight protocols like MQTT, Examon efficiently transmits data at high frequencies, which is essential for large-scale HPC environments, and this utilization with simulation data. Visualization is facilitated through Grafana, providing interactive dashboards that enable in-depth analysis of system behavior, performance trends, and energy consumption.

Due to the restricted capabilities of the TPC hardware instantiated on the PL of the FPGA and the limited computation power of the FPGA PS cores, both the simulation and TPC firmware operate at 200× slower than real-time. Specifically, a slowdown factor of 25× is attributed to the reduced PLL frequency of the TPC hardware, which operates below the achievable $500MHz$ of the actual hardware due to FPGA limitations [110]. An additional 8× slowdown is introduced by the simulation running on the PS cores to prevent deadline misses, with a built-in check to verify deadline adherence.

A representation of the HIL Co-design Framework is depicted in Figure 6.2.

6.5 Tests and Results on the HIL Co-design Framework

In this section, a series of tests and experimental studies conducted in collaboration with other colleagues using the HIL co-design framework are presented. The platform's comprehensive capabilities enabled precise investigation across multiple research objectives, facilitating analyses under varying conditions. Each experiment

leveraged the platform's architecture, underscoring its utility in testing and prototyping. Collectively, these studies demonstrate the platform's effectiveness as a reliable foundation for experimentation in high-performance environments.

6.5.1 Control Algorithm Execution Timing

In [111], the Hardware-Software Co-design HIL Framework was used to measure the computational utilization of the BA control algorithm and evaluate the hardware. The authors investigated the speedup provided by the ControlPULP accelerator across various stages of BA, observing improvements ranging from 3× to 7.9× on the 8-core cluster, with the TH-PID and Conv2F control steps demonstrating the best improvement (Figures 3.a and 3.b). This speedup enabled the control of up to 256 HPC cores within a $t_s = 500\mu s$ window.

Additionally, the study analyzed DMA performance and NoC latency delays for ISRs, which provided insights into the data handling and responsiveness capabilities of the ControlPULP hardware. Despite the focus on the hardware, the tests in [111] were evaluated in coordination with the firmware and the control, effectively demonstrating the utility of the Hardware-Software Co-design approach to ensure comprehensive validation within the HIL environment. This setup underscores the capability of ControlPULP to handle complex control tasks efficiently and in real-time, offering a viable path forward for advanced control methodologies in HPC systems.

6.5.2 SCMI Communication Testing

The System Control and Management Interface (SCMI) is a standardized protocol introduced by ARM to manage power and performance management communication between HLC and LLC controllers in HPC environments [92]. Originally developed to overcome the limitations of OS-centric power management, SCMI addresses the need for real-time, low-latency communication through a standardized interface that allows HLCs to send control target requirements to dedicated LLCs. This delegation model not only reduces overhead on application-class processors but also enhances response times for critical control actions, making SCMI an effective protocol for DPTM in systems where conditions vary rapidly.

The hardware-software co-design framework developed in this work, enabled the implementation and testing of SCMI, providing a platform where the interaction between HLC and LLC could be examined under realistic conditions. This approach allowed the authors in [159] to test SCMI's latency, reliability, and effectiveness in managing power control commands and DVFS adjustments, while assessing the impact of communication delays on the control requirements of HPC systems. SCMI was implemented using a dedicated mailbox-based communication channel between the HLC and LLC, which operates via shared memory. This setup allowed efficient data transfer and control signaling, where each control request from the HLC is passed to the LLC through interrupt-driven notifications.

Results from [159] demonstrated that SCMI does not disrupt power management policies within an HPC control environment, achieving a message dispatch time through the Linux software stack of 70.5μs on average and an LLC processing response time averaging 603.5μs. Although relatively high, this value also accounts for latencies introduced by the real-time scheduling of the communication task in the TPC firmware, which operates with a 1ms interval as outlined in section 6.3.2. Indeed, results show a uniform distribution ranging from 83μs to 1065μs, with occasional outliers, as shown in Figure 3 of [159]. When isolating the latency specifically from

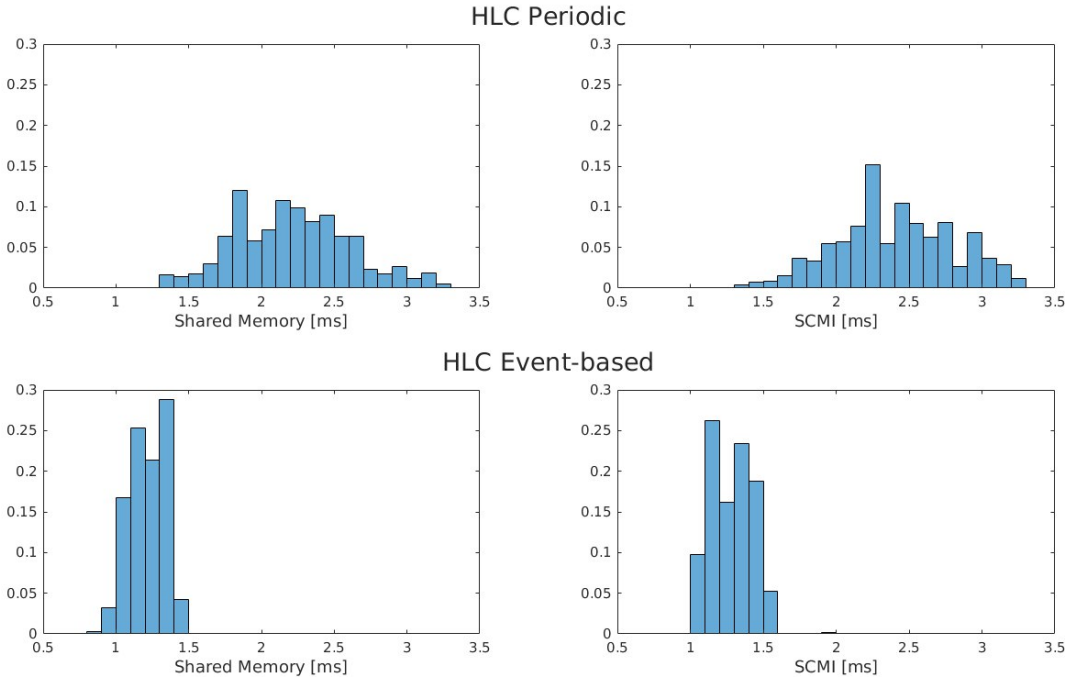


FIGURE 6.3: Comparison between SCMI mailbox communication and shared DRAM, used as a reference, in both periodic and event-based HLC configurations.

the mailbox, interrupt handling, associated ISR, and the return to execution, the total SCMI management time is reduced to 13.15μs, as reported in Table I of [159].

Figure 6.3 [1] illustrates the measured latency between a workload phase transition—specifically, shifting from memory-bound to high-power vectorized instructions—and the corresponding adjustment of the operating point by the LLC. The study examines the comparison between the shared DRAM data exchange, which serves as a baseline with minimal inherent hardware latency, alongside the SCMI communication implementation for providing DVFS targets to the LLC. The analysis includes both a periodic HLC version and an event-triggered version that activates upon detecting a workload phase edge change. Results indicate that, on average, SCMI introduces no significant delay to this process, with total latencies around 2.25 ms for the periodic configuration and 1.15 ms for the event-triggered configuration—values that align well with the considerations discussed in sections 3.3.4 and 6.3.2.

This comparison highlighted SCMI's ability to handle the communication demands commonly encountered in HPC workloads, ensuring timely application of control policies without introducing significant delays. Overall, the hardware-software co-design framework not only facilitated a reliable SCMI implementation but also enabled extensive performance characterization, confirming SCMI's suitability for HPC environments.

6.5.3 EPI Light Reference Platform (LRP)

To further evaluate the TPC, the EPI project provided a Light Reference Platform (LRP) for testing the controller alongside real VRM hardware. The Hardware-Software Co-design framework, utilizing FMC connection cables, enabled interfacing of the

[1] Replication of Figure 4 from [159].

FIGURE 6.4: The EPI LRP (on the right) connected to the FPGA (on the left) executing the HIL framework.

TPC software stack—including the control action—with both the VRMs and the BMC on the LRP. This setup highlights the versatility of the co-design platform and its ability to interact with real hardware components, thereby enhancing the validity of the HIL testing environment. Such adaptability in testing both simulated and physical hardware reinforces the developed platform's value testing setup.

6.6 Conclusion

The Hardware in the Loop (HIL) co-design framework built in this work is a pivotal tool in the development and validation of the Thermal and Power Controller (TPC), providing a seamless, iterative approach to implementing and testing it across all its layers: hardware, firmware, and control algorithms. Unlike MIL, this framework allows an authentic deployment of the TPC enabling a detailed study of the interaction between its components and an evaluation of its realistic performance.

By integrating a real embedded controller with an HPC system simulation, it allows for rapid evaluation, refinement, and benchmarking of design changes. This enables a continuous improvement process, where the impact of each modification—whether in hardware design, firmware optimizations, or control strategy tuning—can be directly tested, tracked, and validated.

The co-design framework is structured around three key components:

- **Simulated HPC Environment**: a software model of the computing chiplet, incorporating thermal, power, and performance models derived in previous chapters. This model interacts with the TPC in real-time through shared memory, emulating the behavior of a real HPC system under dynamic workloads.

- **Embedded TPC Controller**: a real implementation of the TPC with its hardware and firmware. On it, the control algorithms described in Chapter 4 run in a real-time environment, interfacing with the simulated system as if it were controlling an actual HPC processor.
- **Monitor Interface**: through Examon a logging and data collection system that tracks control performance, power and thermal system response. This facilitates quantitive comparison and qualitative visual validation.

The HIL co-design framework has been utilized in multiple research projects, demonstrating its effectiveness as a validation platform for TPC. One key application involved evaluating the execution timing and computational utilization of the BA control algorithm, and develop an optimized parallel version that runs on the ControlPULP cluster. Another important use case focused on testing the SCMI communication protocol for low-latency power and performance management, where the framework enabled direct evaluation of message dispatch times and LLC response latencies, ensuring compliance with real-time requirements.

Additionally, the framework facilitated integration with physical hardware components, as demonstrated by the testing of the EPI Light Reference Platform (LRP), where the TPC controller was tested alongside real VRMs and management interfaces. This capability highlights the versatility of the framework in supporting both simulated and physical hardware testing, reinforcing its value as a continuous integration (CI) environment.

Chapter 7

Conclusion

This thesis tracks a journey through the intricate landscape of power and thermal control in HPC systems, collecting several years of research into this evolving field. Leveraging the collaboration and participation in the European Project Initiative (EPI) project, the work was structured to establish solid foundations, beginning with a detailed system description, followed by an in-depth analysis of the key control challenges, and culminating in the design, implementation, and comparison of advanced control algorithms. All the necessary steps to develop a Thermal and Power Controller (TPC) are comprised in this study, from modeling the system and defining control strategies to setting up both Model in the Loop (MIL) and Hardware in the Loop (HIL) testing frameworks to enable validation.

Clarifying the objectives of the TPC by distinguishing the roles of each control layer and describing more in detail the system elements involved in regulation with their non-idealities, are critical parts of this work. Research in this field has lately been constrained by outdated assumptions and misconceptions about modern CPU architectures, resulting in incomplete control methodologies that overlook key limitations and interactions within modern many-core designs, and confusion regarding the specific responsibilities of each cascade control layer.

At its core, this research is about understanding and mastering complexity. It began by developing a comprehensive system model, designed specifically for control-oriented simulations, integrating thermal behavior, power consumption dynamics, and performance modeling. Unlike traditional models focused on architectural and hardware design, this simulation was built to capture the key constraints, actuator limitations, and non-idealities relevant to low-level power and thermal control, ensuring that control strategies could be tested under realistic, dynamically changing conditions.

With this foundation, the study moved towards the development and comparison of control algorithms, progressing from industry-inspired baseline techniques to advanced fuzzy control and predictive Model Predictive Control (MPC) strategies. The results demonstrated that traditional PID-based methods struggle with modern many-core constraints, while iterative fuzzy control offers a robust solution to managing power and temperature effectively.

MPC, while theoretically well-suited for handling multi-variable constraints and optimization, encounters computational challenges that complicate its deployment in embedded Low-Level Controllers (LLCs). To meet real-time execution constraints, simplified and linearized versions must be employed, limiting its ability to fully exploit its predictive mode-based capabilities. As a result, MPC struggles to consistently outperform heuristic and simpler approaches in the fast-reacting, resource-constrained LLC environment. Conversely, it remains a promising option for High-Level Controller (HLC), where computational resources and longer decision horizons

are available, and on the lower layer LLC has already smoothed system response and filtered out peaks and oscillations.

Beyond the algorithms themselves, this work introduced a Hardware in the Loop (HIL) co-design framework, enabling real-world evaluation. This framework allows for iterative design and testing of all aspects of the TPC, including hardware, firmware, and control algorithms, ensuring that each component can be refined and tested under realistic conditions. This contribution is particularly relevant in the rapidly expanding landscape of RISC-V projects, where new processors and architectures are being actively developed. The proposed framework provides a unique opportunity to study TPC implementations tailored to these emerging designs, allowing researchers to explore novel control techniques and validate them on real hardware.

7.1 Looking Forward

This research has demonstrated that while traditional control methodologies may remain effective in some scenarios, emerging architectural challenges require rethinking and refining existing approaches. For instance, MPC benefits are less pronounced in LLC, where power fluctuations, actuator non-idealities, and unpredictable workload noise hinder its prediction capability.

A central challenge in LLC algorithm design is the inability to predict these power peaks and workload shifts, limiting the effectiveness of control strategies. Given the high amplitude and frequency of workload noise, even the best control algorithms struggle to preemptively adjust for such sudden changes. Future research should explore the role of machine learning techniques, such as transformers, in predicting workload transitions and power demands, providing a more informed basis for control decisions. However, any ML-driven approach must be evaluated not only for its prediction accuracy but also for its computational feasibility, as control loop timing constraints remain stringent.

At the hardware level, instruction throttling mechanisms emerge as a critical but underexplored tool for managing these power fluctuations. These mechanisms can selectively delay high-power instructions, helping to enforce power limits set by TPC. However, research into control algorithms that regulate these mechanisms without introducing performance penalties—and in coordination with broader thermal and power management strategies—remains largely absent.

From a control perspective, a dedicated hardware mechanism for accurately measuring per-core power consumption would be highly beneficial. Such sensor could reduce reliance on power models within control algorithms, eliminating the need for indirect estimations and improving the precision of power allocation and thermal management strategies.

A deeper investigation into the interaction and coordination between multiple control components could unlock significant improvements in power and thermal management. The control problem analyzed in this work involves conflicting objectives, such as maximizing execution performance while enforcing power and thermal constraints, alongside requirements spanning multiple metrics, including temperature, power, and current. Additionally, various actuation mechanisms, such as throttling, frequency or voltage scaling, and power gating, provide different means of adjusting the system's operating point. In state-of-the-art approaches—as well as in the control designs presented in this work, with the exception of MPC—these challenges are typically addressed using separate algorithms, each dedicated to a

specific objective, constraint, or even actuator, often structured in parallel or cascade designs. However, achieving a well-structured and harmonized coordination strategy, where each algorithm compensates for the limitations of others without interfering or conflicting, could ensure that no performance potential is left unutilized. A promising direction for future research on this is the development of heuristic control algorithms designed for systems operating over a finite alphabet, or multi-branch tree control schemes. These approaches could provide a more effective decision-making mechanism, handling actuator quantization and coupled constraints.

Another promising direction is distributed control, which is increasingly relevant given the scalability challenges of modern many-core architectures. As CPUs continue to scale up in core count and power density, relying on a single central controller becomes impractical. Instead, distributed local controllers will need to coordinate and negotiate power and thermal constraints dynamically. However, such approaches introduce overhead in communication, delays in consensus, and potentially overconservative constraints that limit achievable performance. Future work should focus on relaxing these constraints intelligently, determining acceptable delays and coordination mechanisms that balance responsiveness with system-wide efficiency.

More broadly, this work can provide insights into chip design in academic research. Decisions regarding sensor placement, voltage domain configurations, and actuator characteristics should be made not just for performance, but controllability in mind. Future architectures can greatly benefit from tighter integration between hardware design and control strategies, ensuring that power and thermal regulation are not just an afterthought, but a fundamental part of system architecture.

7.2 Final Thoughts

Power and thermal control will only become more critical as HPC systems scale in complexity. This research does not claim to provide a definitive solution, but rather a set of refined tools and insights to advance the research in the field forward. By releasing the co-design framework, the hardware, and the firmware as open-source, this work aims to spark new interest and research, enabling both academic and industrial communities to build upon a solid power management groundwork.

With open-source hardware gaining traction, there is a unique opportunity to develop customized control solutions for next-generation computing architectures. The open-source approach enables seamless integration into both research and industrial projects, particularly for emerging RISC-V chips that require advanced thermal and power management strategies.

In that sense, this thesis is not just a conclusion—it is a starting point.

Bibliography

[1] Rahul Agarwal et al. "3D Packaging for Heterogeneous Integration". In: *2022 IEEE 72nd Electronic Components and Technology Conference (ECTC)*. 2022, pp. 1103–1107. DOI: 10.1109/ECTC51906.2022.00178.

[2] Hamdi E. Ahmed et al. "Optimization of thermal design of heat sinks: A review". In: *International Journal of Heat and Mass Transfer* 118 (2018), pp. 129–153. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2017.10.099. URL: https://www.sciencedirect.com/science/article/pii/S0017931017331976.

[3] Dong H. Ahn et al. "Flux: A Next-Generation Resource Management Framework for Large HPC Centers". In: *2014 43rd International Conference on Parallel Processing Workshops*. 2014, pp. 9–17. DOI: 10.1109/ICPPW.2014.15.

[4] Alessandro Alla, Maurizio Falcone, Luca Saluzzi, et al. "A tree structure algorithm for optimal control problems with state constraints". In: *RENDICONTI DI MATEMATICA E DELLE SUE APPLICAZIONI* 41.3-4 (2020), pp. 193–221.

[5] Nicholas Allec et al. "ThermalScope: Multi-Scale Thermal Analysis for Nanometer-Scale Integrated Circuits". In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. 2008, pp. 603–610. DOI: 10.1109/ICCAD.2008.4681639.

[6] Mauricio Altieri Scarpato. "Digital circuit performance estimation under PVT and aging effects". Theses. Université Grenoble Alpes, Dec. 2017. URL: https://theses.hal.science/tel-01773745.

[7] AnandTech Staff. *The AMD Ryzen 7 7800X3D Review: A Simpler Slice of V-Cache for Gaming*. https://www.anandtech.com/show/18795/the-amd-ryzen-7-7800x3d-review-a-simpler-slice-of-v-cache-for-gaming/11. Accessed: 2024.

[8] Arm. *SCP-firmware - version 2.13*. https://github.com/Arm-software/SCP-firmware. 2023.

[9] *Arm© Neoverse™ V2 Core Technical Reference Manual 5.5.1*. ARM. 2022. URL: https://developer.arm.com/documentation/102375/latest/.

[10] Orlando Arrieta, Ramon Vilanova, and Pedro Balaguer. "Procedure for cascade control systems design: choice of suitable PID tunings". In: *International Journal of Computers Communications & Control* 3.3 (2008), pp. 235–248.

[11] Krste Asanović and David A Patterson. "Instruction sets should be free: The case for risc-v". In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146* (2014).

[12] Khaled M. Attia, Mostafa A. El-Hosseini, and Hesham A. Ali. "Dynamic power management techniques in multi-core architectures: A survey study". In: *Ain Shams Engineering Journal* 8.3 (2017), pp. 445–456. ISSN: 2090-4479. DOI: https://doi.org/10.1016/j.asej.2015.08.010. URL: https://www.sciencedirect.com/science/article/pii/S2090447915001380.

[13] Ahmad Taher Azar and Fernando E Serrano. "Design and modeling of anti wind up PID controllers". In: *Complex system modelling and control through intelligent soft computations*. Springer, 2014, pp. 1–44.

[14] Eberhard Baer et al. "Simulation of process variations in FinFET transistor patterning". In: *2016 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*. 2016, pp. 299–302. DOI: 10.1109/SISPAD.2016.7605206.

[15] Giovanni Bambini et al. "An Open-Source Scalable Thermal and Power Controller for HPC Processors". In: *2020 IEEE 38th International Conference on Computer Design (ICCD)*. 2020, pp. 364–367. DOI: 10.1109/ICCD50377.2020.00067.

[16] Giovanni Bambini et al. "Modeling and Controlling Many-Core HPC Processors: an Alternative to PID and Moving Average Algorithms". In: *ACM Trans. Auton. Adapt. Syst.* (Sept. 2024). Just Accepted. ISSN: 1556-4665. DOI: 10.1145/3694687. URL: https://doi.org/10.1145/3694687.

[17] Giovanni Bambini et al. "Modeling the Thermal and Power Control Subsystem in HPC Processors". In: *2022 IEEE Conference on Control Technology and Applications (CCTA)*. 2022, pp. 397–402. DOI: 10.1109/CCTA49430.2022.9966082.

[18] Richard Barry. "FreeRTOS reference manual". In: *Real Time Engineers Ltd* 48 (2011), pp. 88–89.

[19] Richard Barry. "Mastering the FreeRTOS real time kernel". In: *Real Time Engineers Ltd* (2016).

[20] Andrea Bartolini and Davide Rossi. "Advances in power management of many-core processors". In: *Many-Core Computing: Hardware and Software* (2019), p. 191.

[21] Andrea Bartolini et al. "Monte Cimone: Paving the Road for the First Generation of RISC-V High-Performance Computers". In: *2022 IEEE 35th International System-on-Chip Conference (SOCC)*. 2022, pp. 1–6. DOI: 10.1109/SOCC56010.2022.9908096.

[22] Andrea Bartolini et al. "Self-Aware Thermal Management for High-Performance Computing Processors". In: *IEEE Design & Test* 35.5 (2018), pp. 28–35. DOI: 10.1109/MDAT.2017.2774774.

[23] Andrea Bartolini et al. "Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller". In: *IEEE Transactions on Parallel and Distributed Systems* 24.1 (2013), pp. 170–183. DOI: 10.1109/TPDS.2012.117.

[24] Anton Beloglazov et al. "Chapter 3 - A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems". In: ed. by Marvin V. Zelkowitz. Vol. 82. Advances in Computers. Elsevier, 2011, pp. 47–111. DOI: https://doi.org/10.1016/B978-0-12-385512-1.00003-7. URL: https://www.sciencedirect.com/science/article/pii/B9780123855121000037.

[25] Abdelhalim Bendali and Yves Audet. "A 1-V CMOS Current Reference With Temperature and Process Compensation". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.7 (2007), pp. 1424–1429. DOI: 10.1109/TCSI.2007.900176.

[26] F. Beneventi et al. "An Effective Gray-Box Identification Procedure for Multicore Thermal Modeling". In: *IEEE Transactions on Computers* 63.5 (2014), pp. 1097–1110.

[27] Francesco Beneventi et al. "Cooling-aware node-level task allocation for next-generation green HPC systems". In: *2016 International Conference on High Performance Computing & Simulation (HPCS)*. 2016, pp. 690–696. DOI: 10.1109/HPCSim.2016.7568402.

[28] Florian Berberich et al. "European HPC Landscape". In: *2019 15th International Conference on eScience (eScience)*. 2019, pp. 471–478. DOI: 10.1109/eScience.2019.00062.

[29] Srikant Bharadwaj et al. "Predict; Don't React for Enabling Efficient Fine-Grain DVFS in GPUs". In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4*. ASPLOS '23. Vancouver, BC, Canada: Association for Computing Machinery, 2024, 253–267. ISBN: 9798400703942. DOI: 10.1145/3623278.3624756. URL: https://doi.org/10.1145/3623278.3624756.

[30] Ganapati Bhat et al. "Algorithmic Optimization of Thermal and Power Management for Heterogeneous Mobile Platforms". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2017). DOI: 10.1109/TVLSI.2017.2770163.

[31] Nathan Binkert et al. "The gem5 simulator". In: *SIGARCH Comput. Archit. News* 39.2 (Aug. 2011), 1–7. ISSN: 0163-5964. DOI: 10.1145/2024716.2024718. URL: https://doi.org/10.1145/2024716.2024718.

[32] *BIOS and Kernel Developer's Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors*. AMD. 2013.

[33] Åke Björck. *Numerical methods for least squares problems*. SIAM, 2024.

[34] Andrea Borghesi, Alessio Burrello, and Andrea Bartolini. "ExaMon-X: A Predictive Maintenance Framework for Automatic Monitoring in Industrial IoT Systems". In: *IEEE Internet of Things Journal* 10.4 (2023), pp. 2995–3005. DOI: 10.1109/JIOT.2021.3125885.

[35] Björn B. Brandenburg and Mahircan Gül. "Global Scheduling Not Required: Simple, Near-Optimal Multiprocessor Real-Time Scheduling with Semi-Partitioned Reservations". In: *2016 IEEE Real-Time Systems Symposium (RTSS)*. 2016, pp. 99–110. DOI: 10.1109/RTSS.2016.019.

[36] David Brooks, Vivek Tiwari, and Margaret Martonosi. "Wattch: a framework for architectural-level power analysis and optimizations". In: *Proceedings of the 27th Annual International Symposium on Computer Architecture*. ISCA '00. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2000, 83–94. ISBN: 1581132328. DOI: 10.1145/339647.339657. URL: https://doi.org/10.1145/339647.339657.

[37] Nick Brown et al. "Is RISC-V ready for HPC prime-time: Evaluating the 64-core Sophon SG2042 RISC-V CPU". In: *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*. SC-W '23. Denver, CO, USA: Association for Computing Machinery, 2023, 1566–1574. ISBN: 9798400707858. DOI: 10.1145/3624062.3624234. URL: https://doi.org/10.1145/3624062.3624234.

[38] Francesco Bullo et al. *Lectures on network systems*. Vol. 1. CreateSpace, 2018.

[39] Thomas Burd et al. ""Zeppelin": An SoC for Multichip Architectures". In: *IEEE Journal of Solid-State Circuits* 54.1 (2019), pp. 133–143. DOI: 10.1109/JSSC.2018.2873584.

[40] Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis*. Cengage learning, 2015.

[41] Edward A. Burton et al. "FIVR — Fully integrated voltage regulators on 4th generation Intel®Core™SoCs". In: *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*. 2014, pp. 432–439. DOI: 10.1109/APEC.2014.6803344.

[42] Ermao Cai and Diana Marculescu. "Temperature Effect Inversion-Aware Power-Performance Optimization for FinFET-Based Multicore Systems". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.11 (2017), pp. 1897–1910. DOI: 10.1109/TCAD.2017.2666721.

[43] Ermao Cai et al. "Learning-Based Power/Performance Optimization for Many-Core Systems With Extended-Range Voltage/Frequency Scaling". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.8 (2016), pp. 1318–1331. DOI: 10.1109/TCAD.2015.2504330.

[44] Kun Cao et al. "A survey of optimization techniques for thermal-aware 3D processors". In: *Journal of Systems Architecture* 97 (2019), pp. 397–415. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2019.01.003. URL: https://www.sciencedirect.com/science/article/pii/S138376211830540X.

[45] D. Cesarini et al. "COUNTDOWN: a Run-time Library for Performance-Neutral Energy Saving in MPI Applications". In: *IEEE Transactions on Computers* (2020), pp. 1–1.

[46] Chang-Chih Chen and Linda Milor. "Microprocessor Aging Analysis and Reliability Modeling Due to Back-End Wearout Mechanisms". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.10 (2015), pp. 2065–2076. DOI: 10.1109/TVLSI.2014.2357756.

[47] Hsiang-Yun Cheng et al. "Core vs. uncore: The heart of darkness". In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2015, pp. 1–6. DOI: 10.1145/2744769.2647916.

[48] Yingnan Cui, Wei Zhang, and Bingsheng He. "A Variation-Aware Adaptive Fuzzy Control System for Thermal Management of Microprocessors". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.2 (2017), pp. 683–695. DOI: 10.1109/TVLSI.2016.2596338.

[49] Pawel Czarnul, Jerzy Proficz, and Adam Krzywaniak. "Energy-Aware High-Performance Computing: Survey of State-of-the-Art Tools, Techniques, and Environments". In: *Scientific Programming* 2019 (Apr. 2019), pp. 1–19. DOI: 10.1155/2019/8348791.

[50] Mark L. Darby and Michael Nikolaou. "MPC: Current practice and challenges". In: *Control Engineering Practice* 20.4 (2012). Special Section: IFAC Symposium on Advanced Control of Chemical Processes - ADCHEM 2009, pp. 328–342. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2011.12.004. URL: https://www.sciencedirect.com/science/article/pii/S0967066111002528.

[51] Shidhartha Das, Paul Whatmough, and David Bull. "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)*. 2015, pp. 146–151. DOI: 10.1109/ISLPED.2015.7273505.

[52] Anant Deval, Avinash Ananthakrishnan, and Craig Forbell. "Power management on 14 nm Intel® Core M processor". In: *2015 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS XVIII)* (2015), pp. 1–3. URL: https://api.semanticscholar.org/CorpusID:37333321.

[53] Xianzhong Ding et al. "HPC-GPT: Integrating Large Language Model for High-Performance Computing". In: *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*. SC-W '23. Denver, CO, USA: Association for Computing Machinery, 2023, 951–960. ISBN: 9798400707858. DOI: 10.1145/3624062.3624172. URL: https://doi.org/10.1145/3624062.3624172.

[54] Roberto Diversi et al. "Bias-Compensated Least Squares Identification of Distributed Thermal Models for Many-Core Systems-on-Chip". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.9 (2014), pp. 2663–2676. DOI: 10.1109/TCSI.2014.2312495.

[55] Ronald G. Dreslinski et al. "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits". In: *Proceedings of the IEEE* 98.2 (2010), pp. 253–266. DOI: 10.1109/JPROC.2009.2034764.

[56] Lawrence C. Evans. *Partial differential equations*. Providence, R.I.: American Mathematical Society, 2010. ISBN: 9780821849743 0821849743.

[57] Qiu Fang et al. "Thermal-Aware Energy Management of an HPC Data Center via Two-Time-Scale Control". In: *IEEE Transactions on Industrial Informatics* 13.5 (2017), pp. 2260–2269. DOI: 10.1109/TII.2017.2698603.

[58] Gene Franklin, J.D. Powell, and M.L. Workman. *Digital Control of Dynamic Systems-Third Edition*. Nov. 2022. ISBN: ISBN: 0-9791226-3-5 or ISBN13: 978-0-9791226-3-7.

[59] Jessie Frazelle. "Opening up the Baseboard Management Controller: If the CPU is the brain of the board, the BMC is the brain stem." In: *Queue* 17.5 (Jan. 2020), 5–12. ISSN: 1542-7730. DOI: 10.1145/3371595.3378404. URL: https://doi.org/10.1145/3371595.3378404.

[60] Bolin Gao and Lacra Pavel. *On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning*. 2018. arXiv: 1704.00805 [math.OC].

[61] Carlos E. García, David M. Prett, and Manfred Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), pp. 335–348. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(89)90002-2. URL: https://www.sciencedirect.com/science/article/pii/0005109889900022.

[62] Wilfred Gomes et al. "Ponte Vecchio: A Multi-Tile 3D Stacked Processor for Exascale Computing". In: *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 65. 2022, pp. 42–44. DOI: 10.1109/ISSCC42614.2022.9731673.

[63] Graham Clifford Goodwin, Stefan F Graebe, Mario E Salgado, et al. *Control system design*. Vol. 240. Prentice Hall Upper Saddle River, 2001.

[64] Corey Gough, Ian Steiner, and Winston A. Saunders. *Energy Efficient Servers: Blueprints for Data Center Optimization*. 1st. USA: Apress, 2015. ISBN: 1430266376.

[65] Samuel Greengard. "Will RISC-V revolutionize computing?" In: *Communications of the ACM* 63 (Apr. 2020), pp. 30–32. DOI: 10.1145/3386377.

[66] Programmer Guide. *Intel 64 and IA-32 Architectures Software Developer's Manual*. Intel. 2022. URL: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html.

[67] Udit Gupta et al. "Chasing Carbon: The Elusive Environmental Footprint of Computing". In: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 2021, pp. 854–867. DOI: 10.1109/HPCA51647.2021.00076.

[68] Daniel Hackenberg et al. "An Energy Efficiency Feature Survey of the Intel Haswell Processor". In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 2015, pp. 896–904. DOI: 10.1109/IPDPSW.2015.70.

[69] Jawad Haj-Yahya et al. "A Comprehensive Evaluation of Power Delivery Schemes for Modern Microprocessors". In: *20th International Symposium on Quality Electronic Design (ISQED)*. 2019, pp. 123–130. DOI: 10.1109/ISQED.2019.8697544.

[70] Vinay Hanumaiah, Sarma Vrudhula, and Karam S. Chatha. "Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-Core Processors". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.11 (2011), pp. 1677–1690. DOI: 10.1109/TCAD.2011.2161308.

[71] Tony Hey, Stewart Tansley, Kristin Michele Tolle, et al. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA, 2009.

[72] Chung-Hsun Huang and Wei-Chen Liao. "A High-Performance LDO Regulator Enabling Low-Power SoC With Voltage Scaling Approaches". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.5 (2020), pp. 1141–1149. DOI: 10.1109/TVLSI.2020.2972904.

[73] IBM. *OpenPower OCC*. https://github.com/open-power/occ. 2022.

[74] Zihan Jiang et al. "HPC AI500: A Benchmark Suite for HPC AI Systems". In: *Benchmarking, Measuring, and Optimizing*. Ed. by Chen Zheng and Jianfeng Zhan. Cham: Springer International Publishing, 2019, pp. 10–22. ISBN: 978-3-030-32813-9.

[75] Chaoqiang Jin et al. "A review of power consumption models of servers in data centers". In: *Applied Energy* 265 (2020), p. 114806. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.114806. URL: https://www.sciencedirect.com/science/article/pii/S0306261920303184.

[76] Graham Lawerence William Cross Kafil M. Razeeb Eric Dalton and Anthony James Robinson. "Present and future thermal interface materials for electronic devices". In: *International Materials Reviews* 63.1 (2018), pp. 1–21. DOI: 10.1080/09506608.2017.1296605. eprint: https://doi.org/10.1080/09506608.2017.1296605. URL: https://doi.org/10.1080/09506608.2017.1296605.

[77] Andrew B. Kahng et al. "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration". In: *2009 Design, Automation & Test in Europe Conference & Exhibition*. 2009, pp. 423–428. DOI: 10.1109/DATE.2009.5090700.

[78] Kashif Khan et al. "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 3 (Jan. 2018). DOI: 10.1145/3177754.

[79] Philip A Knight. "The Sinkhorn–Knopp algorithm: convergence and applications". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (2008), pp. 261–275.

[80] Jonathan Koomey et al. "Implications of Historical Trends in the Electrical Efficiency of Computing". In: *IEEE Annals of the History of Computing* 33.3 (2011), pp. 46–54. DOI: 10.1109/MAHC.2010.28.

[81] Mario Kovač et al. "European Processor Initiative (EPI)—An Approach for a Future Automotive eHPC Semiconductor Platform". In: *Electronic Components and Systems for Automotive Applications*. Ed. by Jochen Langheim. Cham: Springer International Publishing, 2019, pp. 185–195. ISBN: 978-3-030-14156-1.

[82] Mario Kovač et al. "How Europe Is Preparing Its Core Solution for Exascale Machines and a Global, Sovereign, Advanced Computing Platform". In: *Mathematical and Computational Applications* 25.3 (2020). ISSN: 2297-8747. DOI: 10.3390/mca25030046. URL: https://www.mdpi.com/2297-8747/25/3/46.

[83] Zhiquan Lai et al. "Latency-Aware Dynamic Voltage and Frequency Scaling on Many-Core Architectures for Data-Intensive Applications". In: *2013 International Conference on Cloud Computing and Big Data*. 2013, pp. 78–83. DOI: 10.1109/CLOUDCOM-ASIA.2013.68.

[84] Scott Lathrop and Thomas Murphy. "High-Performance Computing Education". In: *Computing in Science & Engineering* 10.5 (2008), pp. 9–11. DOI: 10.1109/MCSE.2008.132.

[85] John H Lau. *Chiplet design and heterogeneous integration packaging*. Springer, 2023.

[86] Joseph K. L. Lee et al. "Test-Driving RISC-V Vector Hardware for HPC". In: *High Performance Computing*. Ed. by Amanda Bienz et al. Cham: Springer Nature Switzerland, 2023, pp. 419–432. ISBN: 978-3-031-40843-4.

[87] Seri Lee and Kevin P. Moran. "Constriction/spreading resistance model for electronics packaging". In: 1996. URL: https://api.semanticscholar.org/CorpusID:28843083.

[88] Charles E. Leiserson et al. "There's plenty of room at the Top: What will drive computer performance after Moore's law?" In: *Science* 368.6495 (2020). ISSN: 0036-8075. DOI: 10.1126/science.aam9744. eprint: https://science.sciencemag.org/content/368/6495/eaam9744.pdf. URL: https://science.sciencemag.org/content/368/6495/eaam9744.

[89] Sheng Li et al. "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures". In: *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO 42. New York, New York: Association for Computing Machinery, 2009, 469–480. ISBN: 9781605587981. DOI: 10.1145/1669112.1669172. URL: https://doi.org/10.1145/1669112.1669172.

[90] Tao Li et al. "Chiplet Heterogeneous Integration Technology—Status and Challenges". In: *Electronics* 9.4 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics\-9040670. URL: https://www.mdpi.com/2079-9292/9/4/670.

[91] W. Liu et al. *BSIM3v3.2.2 MOSFET Model Users' Manual*. Tech. rep. UCB/ERL M99/18. EECS Department, University of California, Berkeley, 1999. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1999/3616.html.

[92] Arm Ltd. *Power and Performance Management using Arm SCMI Specification*. Tech. rep. Aug. 2019.

[93] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[94] Ravi Mahajan et al. "Embedded Multi-die Interconnect Bridge (EMIB) – A High Density, High Bandwidth Packaging Interconnect". In: *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*. 2016, pp. 557–565. DOI: 10.1109/ECTC.2016.201.

[95] Matthias Maiterth et al. "Energy and power aware job scheduling and resource management: Global survey—initial analysis". In: *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE. 2018, pp. 685–693.

[96] Abhinandan Majumdar et al. "Dynamic GPGPU Power Management Using Adaptive Model Predictive Control". In: *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2017, pp. 613–624. DOI: 10.1109/HPCA.2017.34.

[97] Sumit K. Mandal et al. "An Energy-aware Online Learning Framework for Resource Management in Heterogeneous Platforms". In: *ACM Trans. Des. Autom. Electron. Syst.* 25.3 (May 2020). ISSN: 1084-4309. DOI: 10.1145/3386359. URL: https://doi.org/10.1145/3386359.

[98] James Martin. "Programming real-time computer systems". In: (1965).

[99] Seda Ogrenci Memik et al. "Optimizing Thermal Sensor Allocation for Microprocessors". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.3 (2008), pp. 516–527. DOI: 10.1109/TCAD.2008.915538.

[100] Franc Mihalič, Mitja Truntič, and Alenka Hren. "Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges". In: *Electronics* 11.15 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11152462. URL: https://www.mdpi.com/2079-9292/11/15/2462.

[101] Kasra Moazzemi et al. "HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management". In: *ACM Trans. Embed. Comput. Syst.* 18.5s (2019). ISSN: 1539-9087. DOI: 10.1145/3358203. URL: https://doi.org/10.1145/3358203.

[102] M. Moudgill, P. Bose, and J.H. Moreno. "Validation of Turandot, a fast processor model for microarchitecture exploration". In: *1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305)*. 1999, pp. 451–457. DOI: 10.1109/PCCC.1999.749471.

[103] Rajeev Muralidhar, Renata Borovica-Gajic, and Rajkumar Buyya. "Energy Efficient Computing Systems: Architectures, Abstractions and Modeling to Techniques and Standards". In: *ACM Comput. Surv.* 54.11s (2022). ISSN: 0360-0300. DOI: 10.1145/3511094. URL: https://doi.org/10.1145/3511094.

[104] Samuel Naffziger et al. "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product". In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 2021, pp. 57–70. DOI: 10.1109/ISCA52012.2021.00014.

[105] Marco A. S. Netto et al. "HPC Cloud for Scientific and Business Applications: Taxonomy, Vision, and Research Challenges". In: *ACM Comput. Surv.* 51.1 (Jan. 2018). ISSN: 0360-0300. DOI: 10.1145/3150224. URL: https://doi.org/10.1145/3150224.

[106] Hung T Nguyen and Michio Sugeno. *Fuzzy systems: modeling and control*. Vol. 2. Springer Science & Business Media, 2012.

[107] Giuseppe Notarstefano, Ivano Notarnicola, Andrea Camisa, et al. "Distributed optimization for smart cyber-physical networks". In: *Foundations and Trends® in Systems and Control* 7.3 (2019), pp. 253–383.

[108] Kenneth O'Neal and Philip Brisk. "Predictive Modeling for CPU, GPU, and FPGA Performance and Power Consumption: A Survey". In: *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 2018, pp. 763–768. DOI: 10.1109/ISVLSI.2018.00143.

[109] Alessandro Ottaviano et al. "An On-Die Multi-core RISC-V Accelerator for MPC-based Energy and Thermal Management of HPC Processors". 2025.

[110] Alessandro Ottaviano et al. "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". In: *International Journal of Parallel Programming* (2024). ISSN: 1573-7640. DOI: 10.1007/s10766-024-00761-4. URL: https://doi.org/10.1007/s10766-024-00761-4.

[111] Alessandro Ottaviano et al. "ControlPULP: A RISC-V Power Controller for HPC Processors with Parallel Control-Law Computation Acceleration". In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by Alex Orailoglu, Marc Reichenbach, and Matthias Jung. Cham: Springer International Publishing, 2022, pp. 120–135. ISBN: 978-3-031-15074-6.

[112] Kenneth O'brien et al. "A Survey of Power and Energy Predictive Models in HPC Systems and Applications". In: *ACM Comput. Surv.* 50.3 (June 2017). ISSN: 0360-0300. DOI: 10.1145/3078811. URL: https://doi.org/10.1145/3078811.

[113] G. Paci et al. "Exploring " temperature-aware " design in low-power MPSoCs". In: *Proceedings of the Design Automation & Test in Europe Conference*. Vol. 1. 2006, pp. 1–6. DOI: 10.1109/DATE.2006.243741.

[114] Seungwook Paek et al. "Area-efficient dynamic thermal management unit using MDLL with shared DLL scheme for many-core processors". In: *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. 2011, pp. 1664–1667. DOI: 10.1109/ISCAS.2011.5937900.

[115] Santiago Pagani et al. "Machine Learning for Power, Energy, and Thermal Management on Multicore Processors: A Survey". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.1 (2020), pp. 101–116. DOI: 10.1109/TCAD.2018.2878168.

[116] Jaehyun Park et al. "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors". In: *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. 2010, pp. 419–424. DOI: 10.1145/1840845.1840938.

[117] Anuj Pathania and Jörg Henkel. "HotSniper: Sniper-Based Toolchain for Many-Core Thermal Simulations in Open Systems". In: *IEEE Embedded Systems Letters* 11.2 (2019), pp. 54–57. DOI: 10.1109/LES.2018.2866594.

[118] Rui Pereira et al. "Ranking programming languages by energy efficiency". In: *Science of Computer Programming* 205 (2021), p. 102609. ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2021.102609. URL: https://www.sciencedirect.com/science/article/pii/S0167642321000022.

[119] A. R. Plummer. "Model-in-the-Loop Testing". In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 220.3 (2006), pp. 183–199. DOI: 10.1243/09596518JSCE207.

[120] Michael J Pont and Royan HL Ong. "Using watchdog timers to improve the reliability of single-processor embedded systems: Seven new patterns and a case study". In: *Proceedings of the First Nordic Conference on Pattern Languages of Programs*. Citeseer. 2002, pp. 159–200.

[121] Simon Portegies Zwart. "The ecological impact of high-performance computing in astrophysics". In: *Nature Astronomy* 4.9 (2020), pp. 819–822.

[122] *Power Control System Architecture - DEN0050*. ARM. 2023. URL: https://developer.arm.com/documentation/den0050/latest/.

[123] Jan M Rabaey. *Digital integrated circuits a design perspective*. 1999.

[124] Md Atiqur Rahman et al. "Advancing thermal management in electronics: a review of innovative heat sink designs and optimization techniques". In: *RSC advances* 14.43 (2024), pp. 31291–31319.

[125] Rajendra K. Raj et al. "High Performance Computing Education: Current Challenges and Future Directions". In: *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '20. Trondheim, Norway: Association for Computing Machinery, 2020, 51–74. ISBN: 9781450382939. DOI: 10.1145/3437800.3439203. URL: https://doi.org/10.1145/3437800.3439203.

[126] Vijay Janapa Reddi et al. "Voltage Smoothing: Characterizing and Mitigating Voltage Noise in Production Processors via Software-Guided Thread Scheduling". In: *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 2010, pp. 77–88. DOI: 10.1109/MICRO.2010.35.

[127] Daniel Reed and Jack Dongarra. "Exascale Computing and Big Data". In: *Communications of the ACM* 58 (July 2015), pp. 56–68. DOI: 10.1145/2699414.

[128] Chiara Irma Riva. "A numerical tool for the analytical solution of temperature rise and thermal spreading resistance for power electronics". In: (2021).

[129] Todd Rosedahl et al. "Power/Performance Controlling Techniques in OpenPOWER". In: *High Performance Computing*. Ed. by Julian M. Kunkel et al. Cham: Springer International Publishing, 2017, pp. 275–289. ISBN: 978-3-319-67630-2.

[130] Davide Rossi et al. "A 60 GOPS/W, -1.8V to 0.9V body bias ULP cluster in 28nm UTBB FD-SOI technology". In: *Solid-State Electronics* 117 (2016). PLANAR FULLY-DEPLETED SOI TECHNOLOGY, pp. 170–184. ISSN: 0038-1101. DOI: https://doi.org/10.1016/j.sse.2015.11.015. URL: https://www.sciencedirect.com/science/article/pii/S0038110115003342.

[131] Davide Rossi et al. "PULP: A parallel ultra low power platform for next generation IoT applications". In: *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE Computer Society. 2015, pp. 1–39.

[132] Efraim Rotem et al. "Power and thermal constraints of modern system-on-a-chip computer". In: *19th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*. 2013, pp. 141–146. DOI: 10.1109/THERMINIC.2013.6675226.

[133] Efraim Rotem et al. "Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge". In: *IEEE Micro* 32.2 (2012), pp. 20–27. DOI: 10.1109/MM.2012.12.

[134] Martin Schlager, Roman Obermaisser, and Wilfried Elmenreich. "A Framework for Hardware-in-the-Loop Testing of an Integrated Architecture". In: vol. 4761. May 2007, pp. 159–170. ISBN: 978-3-540-75663-7. DOI: 10.1007/978-3-540-75664-4_16.

[135] Robert Schöne et al. "Energy Efficiency Features of the Intel Alder Lake Architecture". In: *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*. ICPE '24. London, United Kingdom: Association for Computing Machinery, 2024, 95–106. ISBN: 9798400704444. DOI: 10.1145/3629526.3645040. URL: https://doi.org/10.1145/3629526.3645040.

[136] Robert Schöne et al. "Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance". In: *arXiv:1905.12468 [cs]* (May 2019). arXiv: 1905.12468. URL: http://arxiv.org/abs/1905.12468 (visited on 06/09/2020).

[137] Sébastien Gros et al. "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *International Journal of Control* 93.1 (2020), pp. 62–80. DOI: 10.1080/00207179.2016.1222553.

[138] Siemens Digital Industries Software. *FloTHERM: Thermal Simulation Software*. Siemens Digital Industries Software. 2024. URL: https://plm.sw.siemens.com/it-IT/simcenter/fluids-thermal-simulation/flotherm/.

[139] Rafael Ferreira da Silva et al. "Frontiers in Scientific Workflows: Pervasive Integration With High-Performance Computing". In: *Computer* 57.8 (2024), pp. 36–44. DOI: 10.1109/MC.2024.3401542.

[140] Yogesh Singh Chauhan et al. "BSIM — Industry standard compact MOSFET models". In: *2012 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*. 2012, pp. 46–49. DOI: 10.1109/ESSDERC.2012.6343330.

[141] Kevin Skadron et al. "Temperature-aware microarchitecture". In: *SIGARCH Comput. Archit. News* 31.2 (May 2003), 2–13. ISSN: 0163-5964. DOI: 10.1145/871656.859620. URL: https://doi.org/10.1145/871656.859620.

[142] Kevin Skadron et al. "Temperature-aware microarchitecture: Modeling and implementation". In: *ACM Trans. Archit. Code Optim.* 1.1 (Mar. 2004), 94–125. ISSN: 1544-3566. DOI: 10.1145/980152.980157. URL: https://doi.org/10.1145/980152.980157.

[143] Carlos A Smith and Armando B Corripio. *Principles and practices of automatic process control*. John wiley & sons, 2005.

[144] Bartolomeo Stellato et al. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2. URL: https://doi.org/10.1007/s12532-020-00179-2.

[145] Bo Su et al. "PPEP: Online Performance, Power, and Energy Prediction Framework and DVFS Space Exploration". In: *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. 2014, pp. 445–457. DOI: 10.1109/MICRO.2014.17.

[146] Hameedah Sultan, Anjali Chauhan, and Smruti R. Sarangi. "A Survey of Chip-level Thermal Simulators". In: *ACM Comput. Surv.* 52.2 (Apr. 2019). ISSN: 0360-0300. DOI: 10.1145/3309544. URL: https://doi.org/10.1145/3309544.

[147] Jiming Sun et al. *Embedded Firmware Solutions: Development Best Practices for the Internet of Things*. Springer Nature, 2015.

[148] Synopsys, Inc. *HSPICE: Circuit Simulation and Analysis Software*. Synopsys, Inc. Mountain View, CA, 2024. URL: https://www.synopsys.com/implementation-and-signoff/ams-simulation/primesim-hspice.html.

[149] Zhangxi Tan et al. "A case for FAME: FPGA architecture model execution". In: *Proceedings of the 37th Annual International Symposium on Computer Architecture*. ISCA '10. Saint-Malo, France: Association for Computing Machinery, 2010, 290–301. ISBN: 9781450300537. DOI: 10.1145/1815961.1815999. URL: https://doi.org/10.1145/1815961.1815999.

[150] Andrew Tanenbaum. *Modern operating systems*. Pearson Education, Inc., 2009.

[151] Arun K Tangirala. *Principles of system identification: theory and practice*. CRC press, 2018.

[152] Danielle C. Tarraf, Alexandre Megretski, and Munther A. Dahleh. "A Framework for Robust Stability of Systems Over Finite Alphabets". In: *IEEE Transactions on Automatic Control* 53.5 (2008), pp. 1133–1146. DOI: 10.1109/TAC.2008.923658.

[153] A Tekin et al. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*. Tech. rep. Technical report, PRACE, 2021, 2021.

[154] Shyamkumar Thoziyoor et al. "CACTI 5.1". In: (May 2008).

[155] Andrea Tilli et al. "A two-layer distributed MPC approach to thermal control of Multiprocessor Systems-on-Chip". In: *Control Engineering Practice* 122 (May 2022). ISSN: 09670661. DOI: 10.1016/j.conengprac.2022.105099.

[156] Lloyd N Trefethen and David Bau. *Numerical linear algebra*. SIAM, 2022.

[157] Ankush Varma et al. "Power management in the Intel Xeon E5 v3". In: *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 2015, pp. 371–376. DOI: 10.1109/ISLPED.2015.7273542.

[158] A. Vassighi and M. Sachdev. "Thermal runaway in integrated circuits". In: *IEEE Transactions on Device and Materials Reliability* 6.2 (2006), pp. 300–305. DOI: 10.1109/TDMR.2006.876577.

[159] Antonio del Vecchio et al. "Performance Characterization of Hardware/Software Communication Interfaces in End-to-End Power Management Solutions of High-Performance Computing Processors". In: *Energies* 17.22 (2024). ISSN: 1996-1073. DOI: 10.3390/en17225778. URL: https://www.mdpi.com/1996-1073/17/22/5778.

[160] Hai Wang et al. "Compact Piecewise Linear Model Based Temperature Control of Multicore Systems Considering Leakage Power". In: *IEEE Transactions on Industrial Informatics* 16.12 (2020), pp. 7556–7565. DOI: 10.1109/TII.2019.2960414.

[161] Hai Wang et al. "Leakage-Aware Predictive Thermal Management for Multi-core Systems Using Echo State Network". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.7 (2020), pp. 1400–1413. DOI: 10.1109/TCAD.2019.2915316.

[162] Tianyi Wang et al. "On harmonic fixed-priority scheduling of periodic real-time tasks with constrained deadlines". In: *Proceedings of the 53rd Annual Design Automation Conference*. DAC '16. Austin, Texas: Association for Computing Machinery, 2016. ISBN: 9781450342360. DOI: 10.1145/2897937.2898055. URL: https://doi.org/10.1145/2897937.2898055.

[163] Xiaorui Wang et al. "SHIP: Scalable Hierarchical Power Control for Large-Scale Data Centers". In: *2009 18th International Conference on Parallel Architectures and Compilation Techniques*. 2009, pp. 91–100. DOI: 10.1109/PACT.2009.34.

[164] Zhaoqing Wang et al. "Review, Survey, and Benchmark of Recent Digital LDO Voltage Regulators". In: *2022 IEEE Custom Integrated Circuits Conference (CICC)*. 2022, pp. 01–08. DOI: 10.1109/CICC53496.2022.9772734.

[165] T. Webel et al. "Robust power management in the IBM z13". In: *IBM Journal of Research and Development* 59.4/5 (2015), 16:1–16:12. DOI: 10.1147/JRD.2015.2446872.

[166] Qing Xie, Mohammad Javad Dousti, and Massoud Pedram. "Therminator: a thermal simulator for smartphones producing accurate chip and skin temperature maps". In: *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*. ISLPED '14. La Jolla, California, USA: Association for Computing Machinery, 2014, 117–122. ISBN: 9781450329750. DOI: 10.1145/2627369.2627641. URL: https://doi.org/10.1145/2627369.2627641.

[167] AMD Xilinx. *Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*. https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html. Accessed: 2024-09-12. 2024.

[168] Wei-Bin Yang, Yu-Yao Lin, and Yu-Lung Lo. "Analysis and design considerations of static CMOS logics under process, voltage and temperature variation in 90nm CMOS process". In: *2014 International Conference on Information Science, Electronics and Electrical Engineering*. Vol. 3. 2014, pp. 1653–1656. DOI: 10.1109/InfoSEEE.2014.6946202.

[169] Toshio Yoshida. "Fujitsu high performance CPU for the Post-K Computer". In: *Hot Chips*. Vol. 30. 2018.

[170] Melanie N. Zeilinger et al. "On real-time robust model predictive control". In: *Automatica* 50.3 (2014), pp. 683–694. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.11.019. URL: https://www.sciencedirect.com/science/article/pii/S0005109813005360.

[171] Xunwei Zhou et al. "Investigation of candidate VRM topologies for future microprocessors". In: *IEEE Transactions on Power Electronics* 15.6 (2000), pp. 1172–1182. DOI: 10.1109/63.892832.