Thermal and Energy Management of High-Performance Multicores: Distributed and Self-Calibrating Model-Predictive Controller

Andrea Bartolini, Matteo Cacciari, Andrea Tilli, and Luca Benini, *Fellow*, *IEEE*

Abstract—As result of technology scaling, single-chip multicore power density increases and its spatial and temporal workload variation leads to temperature hot-spots, which may cause nonuniform ageing and accelerated chip failure. These critical issues can be tackled by closed-loop thermal and reliability management policies. Model predictive controllers (MPC) outperform classic feedback controllers since they are capable of minimizing performance loss while enforcing safe working temperature. Unfortunately, MPC controllers rely on a priori knowledge of thermal models and their complexity exponentially grows with the number of controlled cores. In this paper, we present a scalable, fully distributed, energy-aware thermal management solution for single-chip multicore platforms. The model-predictive controller complexity is drastically reduced by splitting it in a set of simpler interacting controllers, each one allocated to a core in the system. Locally, each node selects the optimal frequency to meet temperature constraints while minimizing the performance penalty and system energy. Comparable performance with state-of-the-art MPC controllers is achieved by letting controllers exchange a limited amount of information at runtime on a neighborhood basis. In addition, we address model uncertainty by supporting learning of the thermal model with a novel distributed self-calibration approach that matches well the controller architecture.

Index Terms—Thermal control, energy minimization, multicore, model predictive controller, system identification

1 Introduction

Driven by Moore's law, the trend in increasing performance of CPUs has seen as collateral effects the rapid increase of power consumption and power density that are at the root of performance degradation, acceleration of chip ageing, and cooling costs. Cooling and heat management are rapidly becoming the key limiters for high performance processors, especially for HPC and data centers which typically host clusters of hundreds (and sometimes even thousands) of high-performance processors. Thus, approximately 50 percent of the energy consumed by data centers is used for powering the cooling infrastructure. The remaining energy is used for computation and causes the temperature ramp-up. This trend is globally visible, in 2009 about 2 percent of the global electricity production was consumed to operate data centers worldwide, and accounted for an estimated 0.7 percent of the global energy-related CO2 emissions [1], [2].

The research community and leading electronics companies have invested significant efforts in developing thermal control solutions for computing platforms, limiting the overhead imposed by worst case thermal design in both cost and performance. Indeed, even if design optimization can improve the architectural thermal efficiency [3], on-chip silicon performance and usage show highly spatial and temporal variability; this reflects in the inefficiency of static approaches to solve thermal issues, causing HW damage, reliability loss and cooling costs overhead.

Moreover, safe operation mode in HPC and data center clusters relies on the availability of the cooling infrastructure. Indeed, as it has been recently demonstrated in case of power outage, the cooling system will have a stop whereas the UPS system will provide energy only to racks that will consequently run in a thermal emergency in a range of time between 30 to 400 s [4] accordingly to cabinet power. In addition, if CPU fans are slowed down due to failures or design constraints such as in laptops, the system performance can be severely reduced, below the 60 percent of nominals one [5], [6]. It has been also demonstrated that the CPU heat often flows in the DIMMs inside the blade. This reduces the DIMMs DRAM densities with an estimated impact of the 13 percent loss of the overall performance [7]. As a completely different applications, games are pushing today's desktop and laptop processors for consumer market to the performance limits, and as it can be seen in a large variety of modding/overclocking web forums their gaming experiences is mined by thermal issues.[1]

Thus, the main objective of the dynamic thermal control is to adapt at runtime the resource usage to achieve an optimal tradeoff between performance, quality-of-service (QoS), power consumption, and cooling effort, while at the same time ensuring safe working temperature under all working conditions. Nowadays, multicore processors

- *A. Bartolini and L. Benini are with the Department of Electronics, Computer Sciences and Systems, University of Bologna, via Risorgimento 2, 40136 Bologna, Italy.*
  E-mail: {a.bartolini, luca.benini}@unibo.it.
- *M. Cacciari and A. Tilli are with the Center for Research on Complex Automated Systems (CASY), Department of Electronics, Computer Sciences and Systems, University of Bologna, via Carlo Pepoli 3/2, 40136 Bologna, Italy. E-mail: {matteo.cacciari, andrea.tilli}@unibo.it.*

Manuscript received 26 Aug. 2011; revised 30 Jan. 2012; accepted 22 Mar. 2012; published online 28 Mar. 2012.
Recommended for acceptance by M.E. Acacio.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-08-0570.
Digital Object Identifier no. 10.1109/TPDS.2012.117.

1. http://www.tomshardware.com, http://www.techpowerup.com/realtemp/, http://cpu.rightmark.org/articles.shtml.

include hardware support for dynamic power and thermal management, based on introspective monitors, sensors and performance knobs. This infrastructure provides the sensors and the actuators for feedback control management policies.

1.1 Related Work

Power budgeting and power capping [8] techniques use built-in power meters as inputs to feedback controllers for constraining the power consumption to a given budget by reducing the cores clock frequencies. This approach has two main drawbacks: first, it relies on the availability of accurate and reliable power measurement infrastructures; second, in many cases, it does not lead to a global energy reduction since it usually leads to longer execution times and higher static power. Indeed, high power dissipation phases usually are correlated with CPU-bound computational ones. Thus, power budgeting techniques happen to reduce the frequency mainly in this situation, leading to energy losses due to static power and to execution time linear dependency with frequency [9]. Different solutions have been presented to achieve a system energy reduction, but unfortunately energy minimization alone cannot enforce a safe working temperature [9], [10].

Closed-loop thermal control policies aim to address this problem. Threshold-based thermal control techniques are the most widely used today in both HW and SW [11], [12]. To avoid chip damage, microprocessors automatically shut down with a hardware trigger when core temperature crosses a safe temperature limit. In addition, operating systems exploit HW knobs (core voltage and frequency and power gating) and temperature sensors readings with threshold-based feedback control to avoid emergency thermal shutdown. These techniques have major drawbacks, particularly for multiscale systems such as many-core and 3D-integrated stacks [13], [14], [15]. Hardware-triggered temperature capping brings major performance degradation or even application failure, while OS-based capping cannot safely bound the runtime temperature and it has been shown to worsen the thermal cycles and system reliability [17], [16], [12].

To overcome these limitations, classic feedback controllers have been proposed to adjust hardware knobs smoothly [17], [18], [19]. However, simple feedback controllers, such as proportional integral derivative (PID) controllers are not sufficiently flexible and powerful for the complex multimodal dynamic behavior of many-core heterogeneous SoCs [8]. Recently, Model Predictive Controllers (MPC) have been proposed; they rely on a system model to predict the future temperature while finding the optimal control action by solving a constrained optimization problem for one or more control steps in the future [20], [21], [22], [8]. If an accurate thermal model is available, MPCs can guarantee a reliable temperature capping in any working condition. Indeed, Zanini et al. [23] show a comparison between MPC, Linear Quadratic Regulator (LQR), and Threshold-Based Dynamic Voltage and Frequency Scaling (TB-DVFS). The results highlight that while keeping under control the temperature, the MPC has higher throughput than the LQR and TB-DVFS, which execute, respectively, 25 and 50 percent less workload than the MPC. Wang et al. [8] present a MPC that constraint both the power and the temperature of the cores while maximizing the performance. It uses power sensor as input to the optimization problem and to generate online a frequency-to-power linear model. This reduces the complexity of the controller (even though it can lead to sub-optimal controller corrective actions). Zanini et al. [22] assume a workload input requirement, so that the MPC minimizes the performance loss in tracking it while constraining the core temperatures. Internally, it adopts a nonlinear frequency-to-power model, statically precomputed offline, avoiding usage of power sensors. On the other hand, the adopted model does not consider the power dissipation dependency on workload properties, assuming it to be only related to core frequency. De La Guardia and Beltman [24] use a MPC to control the fan speed to minimize the acoustic noise while adapting it to variable computational workloads. The MPC is shown to be capable of accurately managing both the not-linear relations between the fan speed and the thermal resistance of the heat sink and between the fan speed and the human noise perception. All these relationships are assumed to be precharacterized.

Indeed, performance of MPC solutions strongly depends on the thermal model accuracy and availability. Unfortunately, often an accurate model is not a priori available. Different approaches have been proposed to identify it from HW introspective monitors. In [8], a first-order thermal model is estimated using offline least square method. Differently, Cochran and Reda [25] extract a set of linear models to relate temperatures to workload characteristics and core frequencies. Eguia et al. [26] combine identification techniques with an overfitting remedying algorithm to reduce complexity and improve accuracy; nevertheless, the results need to be improved for fast power changes and thermal interactions between adjacent cores. Kumar and Atienza [27] propose a hardware implementation for Neural Network (NN) simulators to effectively simulate the thermal model of the systems at runtime. The predictive accuracy of the NN can be improved by combining online refinement with the offline training. This approach requires specialized HW and thus is not suitable for general purpose multicore.

All the above solutions exploit centralized control and deal with model identification of the whole die. Their complexity and computational burden increase rapidly with the number of cores. Therefore, applying these solutions in upcoming many-cores [28] is very expensive. This complexity problem has been addressed in the literature. In [18], a distributed thermal control approach based on classical PID control and timing error avoidance scheme is presented. The authors highlight the benefit of adopting a distributed approach instead of a centralized one. On the other hand, this requires to adopt on each core a simpler feedback controller which is not able of predicting the thermal interaction with neighbor cores.

Zanini et al. [29] use a model-order reduction techniques to reduce the complexity of the MPC; the complexity reduction is significant compared with the controller performance loss. This approach is effective in reducing the model order when starting from a finer model of the same floorplan. As the number of cores grows the scenario will be different and the complexity of centralized MPC solution will be driven by the large number of integrated cores even if each one is represented by very simple thermal model. Indeed, as we will show in the experimental results section, even by considering a second-order thermal model the centralized MPC complexity does not scale well as the number of integrated core increases.

Coskun et al. [30] use an autoregressive moving average (ARMA) technique for predicting the future thermal evolution for each core. This model is capable of predicting future temperature only based on previous values. Since they do not account directly for workload-to-power dependency, a Sequential Probability Ratio Test (SPRT) technique is then used to early detect changes in the statistical residual distribution (average, variance) and than to retrain the model when it is no longer accurate.

Mutapcic and Boyd [31] show how to reduce the optimization problem complexity by decoupling it in a set of parallel ones. This approach has been applied to steady-state thermal optimization and thus need to be improved to tackle fast thermal transients.

In this direction, several authors [32], [33] have shown how to reduce significantly the complexity of centralized MPC by using distributed solutions. This approach is well suited for large-scale systems and consists in dividing the control problem into subproblems with lower complexity.

All the presented solutions are based on the implicit assumption that all the state components are directly measurable through thermal sensors [29], [22], [23], [8]. Unfortunately, important effects in the thermal response of a complex processor are directly related to the different materials that surround the silicon die (heat-sink, heat spreader, etc.) and the associated state components are not directly measurable. Recently, some techniques have been presented to estimate in software missing thermal sensors output.

Reda et al. [34] show a technique to obtain the missing temperature by linear interpolation of the surrounding thermal sensors. The weights of the linear interpolation are optimized offline to minimize the estimation error. To be applied, it requires a reference model or a calibration stage with an infrared camera. Moreover, the estimation is an affine transformation of surrounding temperature; thus, it does not account for dynamic effects and for power spatial changes.

Sharifi and Rosing [35] instead use a Kalman filter trained offline on a reference model to estimate the missing temperature measurement. Compared to the previous solution, it uses for the estimation also the power dissipated by each cores improving the estimation during the transient and it is capable to filter out the noise, usually present, in the thermal sensors readings.

1.2 Contributions

In this paper, we present a complete distributed solution that combines energy minimization, MPC-based thermal control, and thermal model self-calibration.

First, it addresses the scalability challenge of complex control solution for large multicore platform. Indeed, according to the workload characteristics of incoming task, each core (usually referred as node) selects locally the minimum frequency (f_{EM_i}) that allows energy saving, preserving performance loss within a tolerable bound. Then, if necessary, each local MPC controller trims the frequency to ensure a safe working temperature. Local controllers jointly optimize global system operation by exchanging a limited amount of information at runtime on a neighborhood basis.

Second, we address model uncertainty by self-calibration: each thermal controller node extracts automatically the local thermal model by applying a set of training stimuli and monitoring the thermal response of the neighborhood area. The distributed controller strategy combined with the distributed thermal model calibration phase allow us to take advantage of the parallelism of the underlying multi-core platform by running different instances of the controller and self-calibration routine in parallel.

Third, we decoupled the linear and nonlinear parts of the predictive model of each local thermal controller. This allows us to consider complex, nonlinear frequency-to-power models while keeping the MPC internal model linear, minimizing its complexity.

Fourth, in our solution, each local thermal controller embeds a Luenberger state-observer to estimate from the core temperature sensors the not-measurable state components. This allows us to use higher order and more accurate prediction model inside the MPC, bringing to significant improvements in the quality of the control with lower overhead compared to a standard Kalman observer. Moreover, the observer uses the local model identified in the self-calibration phase and thus does not incur in error due to not-in-field calibration.

A preliminary version of the work presented here has been published in [36]. The current version of the manuscript enhances the previous version on several aspects: we give a more detailed description of the internal components; we propose an implicit implementation of the thermal controller that does not require the overhead of designing it offline; we extend the experimental results by evaluating the sensitivity to our design on the accuracy of the internal power model, evaluating its scalability up to 48 cores and comparing its performance against a PID controller.

The remainder of this paper is organized as follows: Section 2 introduces the key aspects of power and thermal issues in a multicore scenario. Section 3 describes the building blocks of the proposed solution. In Section 4, the performance of the thermal controller is fully evaluated in a complete use-case scenario by implementing it in a full-system virtual platform. Final conclusions are drawn in Section 5.

2 Background Concepts

In a multicore die, the temperature distribution along the die area depends on the chip floorplan, thermal dissipation solution, and power consumption of the cores. This latter quantity has been shown to be related to the operating point/performance level and workload characteristics, such as instruction type, density, and data locality [37]; thus, the runtime power can be effectively estimated from performance monitors readings [38], [39], [40]. Following this approach, Fig. 1 shows the results of a set of tests performed to quantify the relationship existing between power, frequency, and Clocks-Per-Instruction (CPI) of the running

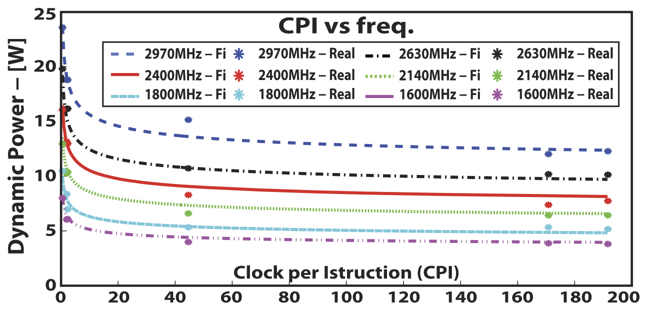


Fig. 1. Per-core power based on activity.

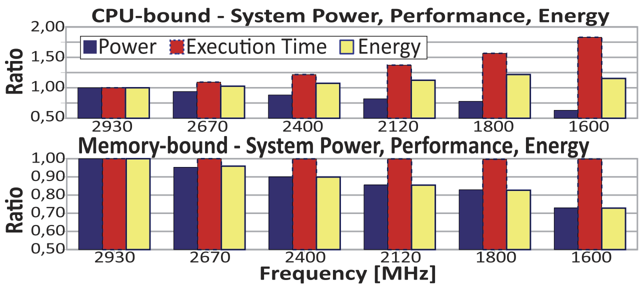


Fig. 2. DVFS "at the wall" power, energy, and performance tradeoff.

task, for each core in a general purpose multicore.[2] The dots represent the actual power consumption, whereas the solid lines represent the fitting model curve extracted by these data, described by

$$P = k_A freq \cdot V_{DD}^2 + k_B + (k_C + k_D freq) \cdot CPI^{k_E}. \quad (1)$$

From (1), we can notice that the core dynamic power depends nonlinearly on the frequency, sublinearly on the CPI of the application and the two dependencies are coupled, CPI dependency is influenced by frequency value. Even if this is a simple, empirical model that does not account for many secondary effects, many works in the state-of-the-art show that it can be used as a reliable basis to develop advanced models to effectively estimate the power consumption of different workloads [40], [38], [39]. Thus, we will use it as reference power model along this paper without any loss of generality. From Fig. 2, we can notice that a significant power reduction can be achieved, not only in CPU-bound program phases (low CPI) but also in memory-bound phases (high CPI). Thus, dynamic voltage and frequency scaling are used to obtain energy efficiency instead of just power reduction. Indeed, whereas scaling the frequency of a core executing a CPU-bound task brings to a performance loss and to a energy inefficiency due to static power and system power, scaling down the frequency of a core executing a memory-bound task does not lead to execution time overhead; thus, the dissipated total energy is reduced.

Leakage power has been shown to nonlinearly depend on silicon temperature; thus, in principle it should be accounted in the power model within the thermal controller. Fortunately, Hanumaiah et al. [41] show that it can be effectively linearized within the range of temperatures typical of the consumers market. Moreover, Hanumaiah et al. [41] show that when power model is used together with a thermal model, this linear approximation in the power model is equivalent to a modification of the thermal model internal parameters. More specifically, if we use a state-space thermal model, this dependency can be mapped in the dynamical matrix coefficients. As we will describe later, our solution learns the thermal model directly from the real hardware; thus, intrinsically the learned parameters contain the contribution of the linearized leakage approximation. In the supplemental online section, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.117, we perform a test to quantify the leakage impact in the Intel SCC multicore [28]. Moreover, as we will see in the experimental results section (4), the thermal controller prevents large temperature variations since it limits at runtime its maximum value. Thus, in this paper, we neglect the static power-temperature dependency in the thermal model used by the controller. Clearly, our experimental platform does model leakage power and its temperature dependency.

To derive a complete thermal model of multicore dies, the "causal chain" $(Frequency, CPI) \rightarrow DissipatedPower \rightarrow Temperature$ can be split in two parts. The first part, as we have seen, is highly nonlinear and can be addressed separately in each core according to (1) or to other nonlinear models, such as those proposed in [40], [38], [39]. Differently, the power-to-temperature model is quasilinear and the temperature variation in each point of the die is affected by the core powers, the current die temperature map and the chip physical properties. This model can be split in simpler interacting models by dividing the die area in regions, aligned with cores for obvious convenience.

It is worth to note that the nonlinear function relating power to frequency is a critical part in the thermal control loop, since uncertainties can cause imprecise control actions. As we mentioned before, we obtained this function by means of empirical experiments but we expect that in future this relation could be provided directly by the manufacturers at the moment of chips release, or it could be learned or updated (to take care of ageing problems) by using online methods. In this last case, we think a power sensor on the chip could be helpful.

According to the granularity of performance counters, thermal and power sensors, our monitoring capability is at core level and within a core we assume uniform power and temperature distributions. Then, recalling Fourier's law, the temperature of each core can be assumed dependent on its own dissipated power, ambient temperature, and adjacent cores temperatures (boundary conditions). This assumption is actually straightforward for continuous time models only. When discrete-time models are considered, a larger coupling among cores has to be considered to account for the "chain of interactions" taking place during the blind intervals among samplings. Recalling again Fourier's law, the coupling among two cores will be inversely related to their distance and directly related to the sampling time period. Hence, the "equivalent neighborhood" of a core depends on the floorplan combined with the adopted sampling period. To verify this assumption, we took an example loosely correlated with the Intel SCC experimental architecture [28]. The floorplan is fully tiled with 48 core/regions, each with an area of $11.82\ \mathrm{mm}^2$ and a maximum power consumption of 2.6 W.

2. Intel Xeon X7350 [42].

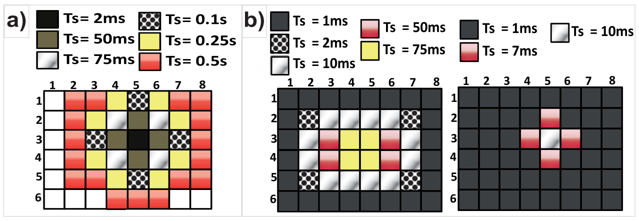


Fig. 3. (a) Single core thermal impact range, at different time windows. (b) Multicores thermal impact range, at different time windows.

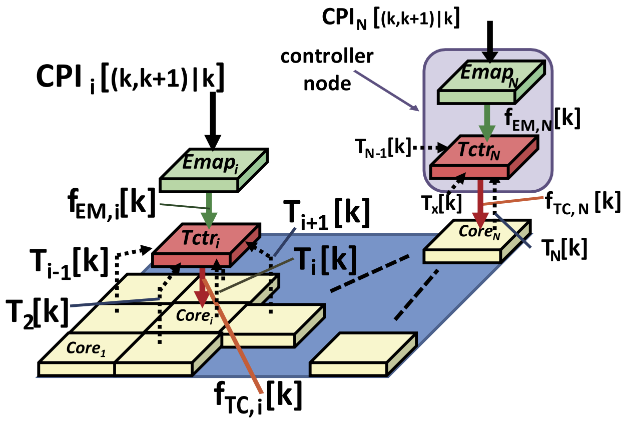


Fig. 4. General architecture.

We used this setup with the HotSpot thermal analysis tool [43], and we made the following test. We stimulate with a power step the central core (5,3) ("thermal attacker" core) while keeping all the other cores at zero power consumption. As result of the power step, the temperature of the attacker core increases, inducing a temperature raise in the neighbor cores. We are interested in measuring the range of thermal impact of the "attacker" core. To do that, we look at the surrounding cores that raise their temperatures as consequence of the attacker. We call them "victims." We consider victim only a core that raises its temperature of 1 percent the attacker temperature increment. Fig. 3a shows in black the attacker core and in different colors the victim after different time intervals. We can notice that the radius of thermal influence of the central core increases with the time interval: within 50 ms, it impacts only the closest core along the four cardinal directions, at 0.75 s the majority of them is affect.

What happens if more cores are triggered with a power step? To answer this question, we made the following test. Considering an increasing numbers of attackers starting from the perimeter cores going close to the victim one (now core 5, 3), after different timing intervals, we check the cores that increase their temperature more than the 1 percent of the attackers increment. In Fig. 3b, we can see the results. We can notice that the neighborhood composed of one core in each direction is enough to prevent the core victim to be sensitive to the rest of the core temperatures within 10 ms of time interval.

Finally, results in [44] highlight that the thermal dynamics of each core is characterized by two time constants: a faster one, at a few ms, is related to the silicon surface, whereas the slower one, at a few seconds, is related to the heat spreader. This behavior needs to be carefully accounted in model identification and control design.

Indeed, all these thermal effects can be modeled using the state-space representation

$$\begin{cases} x[k+1] = A\,x[k] + B\,u[k] \\ y[k] = C\,x[k], \end{cases} \tag{2}$$

where x is the model state vector (the temperature of each elementary component), y is the measured outputs vector (thermal sensors readings), u is the inputs vector (the power dissipation of each core), and A, B, C are, respectively, the dynamical matrix, the input matrix, and the output matrix. In a real system composed of $n-cores$ with one thermal sensor for each core, the y vector has dimension $n \times 1$. To model the behaves of the multimodal thermal transient, we need to use at least a second-order model reflecting in a state vector x with $2n \times 1$ cardinality. If the output matrix C has a unity diagonal, the state vector has a physical meaning: the top $n-components$ represent the silicon core temperatures, whereas the remaining bottom $n-components$ represent the temperature of the different materials in the same spatial region, such as the heat-spreader and heat-sink. It is worth noting that increasing the thermal sensors allows only to probe the directly measurable state components. To the best of our knowledge, the state components associated with the thermal capacitance of nonsilicon-based package materials, such as metallization layers, heat-spreader, and heat-sink are not directly measurable thus need a state observer to be estimated.

In conclusion, the temperature evolution of multicore systems can be effectively modeled decoupling the nonlinear part from the linear part. The first estimates the core power from the runtime workload extracting it from the performance counters and runtime information. The second estimates the core temperature evolution with a linear dynamic model. If we track thermal evolution at a relatively fine time granularity (50 ms or less), this can be done for each core locally considering models with only the neighbor temperatures and core power consumption as input. Moreover, each local thermal model needs to have order higher than 1 to model accurately the multimodal thermal transient. These higher order state components are not directly measurable, requiring a strategy to estimate them.

These conclusions motivate our choice of distributing the thermal control over multiple local controllers, decoupling the linear part to the nonlinear one and embedding a state observer in each of them.

3 Architecture

According to the conclusions drawn in the previous section, for each core a controller node is defined. Fig. 4 depicts the block diagram of the proposed solution. Each controller node (i) is made of three main parts, two of which are executed online and one offline:

- The energy mapper (EM_i): at the kth sampling instant, it takes as input the predicted CPI value of the running task for the following time interval ($CPI_i[[k, k+1]|k]$) and produces as output the core frequency setting ($f_{EM_i}[k]$) for the following time interval $[k,\ k+1]$ that minimizes the core energy consumption, while allowing a tolerable performance overhead.

- The MPC-based thermal controller (TC_i): at the kth time instant, it receives as inputs the Energy Mapper output frequency ($f_{EM_i}[k]$), its own core temperature ($T_i[k]$), the temperatures of the physical neighbors[3] ($T_{NEIGH_i}[k]$), and the ambient temperature ($T_{AMB}[k]$). Then, according to the safe reference temperature (T_{MAX}) at which the core temperatures ($T_i[k]$) must be constrained, the MPC algorithm adjusts the actual frequency command ($f_{TC_i}[k]$), minimizing the displacement from the Energy Mapper requirement.[4]
- The thermal model self-calibration routine: it automatically derives, offline, the local, but interacting, thermal prediction model adopted in MPC-based TC blocks (again according to Section 2).

Section 3.1 describes the Energy Mapper algorithm whereas Section 3.2 describes our thermal controller solution. Section 3.3 instead presents the thermal model self-calibration routine. Section 3.4 describes the implementation of the proposed approach.

3.1 Energy Mapper

The goal of the Energy Mapper is to provide the optimal frequency trajectory ($f_{EM_i}[k]$) to the Thermal Controller (TC_i). Many advanced techniques can be used for this purpose [9], [10]. We choose to select the frequency to minimize the power consumption while preserving the system performance. Our solution does it by taking advantage of the parallel architecture and letting each core (i) compute autonomously the future frequency in line with the incoming workload requirements.

Indeed, considering an in-order architecture,[5] the average time needed to retire an instruction can be seen as composed of two terms: (T_{ALU}) portion of time spent in active cycles and (T_{MEM}) portion of time spent in waiting for memory cycles. Whereas the first term is proportional to the input frequency, the second one is constant to it and depends on the memory access latency.

Assuming f_M the maximum frequency allowed by the system, $f_{CK_i}[k] = \frac{f_M}{\alpha}$ a generic one and given the task CPI requirement for the time instant (k, for the core i ($CPI_i[k]$), we can write the task execution time ($Time_i[k]$) as

$$Time_{M_i}[k] = {}^{\#}INST \cdot [1 + (CPI_i[k] - 1)] \cdot \frac{1}{f_M} \tag{3}$$

$$Time_{CK_i}[k] = {}^{\#}INST \cdot [\alpha + (CPI_i[k] - 1)] \cdot \frac{1}{f_M}. \tag{4}$$

By combining them, the execution time overhead $T_{\%}$ can be represented as function of the new frequency $f_{CK_i[k]}$ and $CPI_i[k]$ as

$$T_{\%,i}[k] = \frac{Time_{CK_i}[k]}{Time_{M_i}[k]} - 1 = \frac{\alpha + (CPI_i[k] - 1)}{1 + (CPI_i[k] - 1)} - 1. \tag{5}$$

Inverting (5), we can find the future frequency ($f_{EM_i}[k]$), output of the Energy Mapper (EM_i), that minimizes the power consumption for the given processor i running a task characterized by $CPI_i[[k, k+1]|k]$, while preserving the performance within a tolerable penalty ($T_{\%}$)

$$f_{EM_i}[k] = \frac{f_M}{1 + CPI_i([k, k+1]|k) \cdot T_{\%}}. \tag{6}$$

3. The sampling time is assumed small enough as discussed in Section 2.
4. The computation and actuation times for EM and TC are assumed negligible with respect to sampling time interval. Hence, for mathematical modeling, control outputs are considered generated at the same instant of sampled inputs.
5. Multicore trend is toward integrating a high number of simpler processors [28].

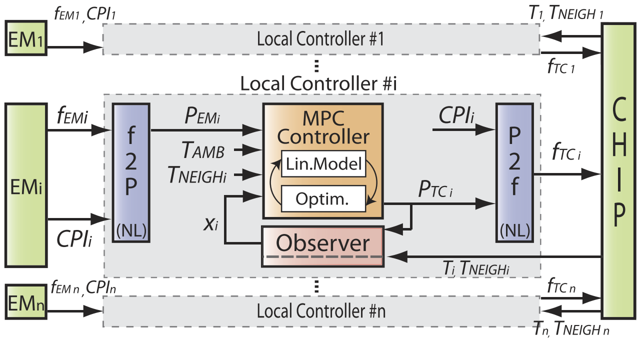


Fig. 5. Thermal controller structure.

3.2 Thermal Controller

In this section, we present our Thermal Controller. In particular, the block diagram in Fig. 5 shows its global architecture focusing on a single local controller, which basically consists of four blocks: f2P converter, P2f converter, MPC Controller, and Observer.

The *f2P converter* block is directly connected to the energy mapper. It takes as inputs the target core speed and the workload, respectively, defined as f_{EM_i} and CPI, and it returns as output the correspondent power consumption (P_{TC_i}). As already introduced in Section 2, this apparently trivial conversion has the important role of encapsulating the nonlinear part of the frequency to temperature relation, enabling the possibility of using a linear MPC controller which guarantees a higher computational efficiency.

The *MPC controller* block represents the center of the local controller. It uses the input measurements T_{AMB}, T_{NEIGH_i} and x_i to predict the core temperature obtained by consuming all the requested powers P_{EM_i}. As output, the block returns the controlled power P_{TC_i} which is equal to P_{EM_i} if the predicted temperature meets the temperature constraint (T_{MAX}), otherwise it is reduced. Clearly, the reduction must be as small as possible to maximize the performance. The MPC approach optimally manages this problem by means of a thermal model of the system,[6] and an optimization problem solver. The model, identified using a self-calibration routine treated in Section 3.3, is used to estimate at each time instant, and starting from the current system temperature, the temperature of the core for a future time window. These predictions are finally used to define the optimization problem:

$$\min_{P_{TC_i}} \sum_{j=0}^{N-1} \| (P_{TC_i}(k+j|k) - P_{EM_i}(k+j)) \|_Q^2 \tag{7}$$

$$\begin{aligned} &\text{subject to} \\ &T_i(k+j|k) \le T_{MAX}, \quad j = 1, \ldots, N-1, \end{aligned} \tag{8}$$

6. In this case, a single core modeled by the system of (2).

where $T_i(k+j|k)$ represents the temperature of the core predicted for time $k+j$ based on the information available at time k. The disequation (8) imposes a hard constraint on the core temperature (T_i), while (7) ensures the maximization of performance, minimizing the difference between the target power (P_{EM_i}) and the power really assigned to the core (P_{TC_i}). N represents the dimension of the prediction time window in sample instants and Q_i is a matrix that weights the importance of the square error elements. In our implementation, we have chosen Q_i equal to the identity matrix and $N = 1$, guided by the characteristics of the system. Each sampling time, the solver yields the optimal solution, P_{TC_i}, that minimizes the cost function and meets the constraints.

The *P2f converter* block is the dual of f2P one. It receives as inputs the optimal P_{TC_i} and the workload of the core, and converts them to a consistent frequency value f_{TC_i}. This optimal frequency is then applied to the core.

Finally, the *Observer* block is used to extract the not measurable state component (x_i) from the temperature measurements. Indeed, in Section 2, the model is shown to have two dynamics, but only one thermal sensor per core is available. By knowing the model and taking as inputs the temperature of the core T_i and P_{TC_i}, the observer estimates the two model state components x_i. Subsequently, this state will be used by the MPC block as initial state for the prediction of the temperature at the next time instant.

More details on blocks implementation will be given in the following sections.

3.2.1 f2P and P2f Converter Blocks

These two blocks perform the conversion, respectively, from frequency to power and from power to frequency using the workload as additional information. The main role of the blocks is to encapsulate the nonlinear part of the frequency to temperature relation. The main advantage of this separation is the possibility of using a linear MPC controller instead of a nonlinear one which allows the use of more efficient and reliable algorithms for computing the optimal control solution. Both the conversion rely on the empirical relation (1). The P2f conversion is obtained inverting (1). Unfortunately, the function is nonlinear, so for finding the root we need to use an iterative numerical method. We have chosen Brent's method [45] that combines the stability of bisection with the speed of a higher order methods. In particular, it uses the secant method or inverse quadratic interpolation if possible and the more robust bisection method if necessary.

3.2.2 MPC Controller Block

In this section, we recall briefly two possible ways to solve the optimization problem presented above. The first is called implicit and provides an iterative algorithm that solves online the optimization problem at each time instant. The second, instead, performs the optimization offline and it is called explicit [21].

To solve the optimization problem described by (7) and (8), both the approaches need for a reformulation of the problem in a standard quadratic programming (QP) form as follows:

$$\min_{s_i} \frac{1}{2} \cdot s_i[k]^T \cdot H_i \cdot s_i[k] + g_i^T \cdot s_i[k] \tag{9}$$

$$\text{subject to} \quad M_i \cdot s_i[k] \leq b_i. \tag{10}$$

This problem is exactly equivalent to the optimization problem previously defined. Whereas $s_i[k]$ is the solution vector, the values of matrices and vectors H_i, M_i, g_i, and b_i can be found starting from (7) and (8). Below are presented the mathematical manipulations to obtain them.

From (7), we find H_i and g_i. The function can be rewritten in the vector form:

$$J = (P_{TC_i} - P_{EM_i})^T \cdot R_i \cdot (P_{TC_i} - P_{EM_i}),$$

where

$$P_{TC_i} = [P_{TC_i}[k|k] \ \dots \ P_{TC_i}[k+i|k] \ \dots \ P_{TC_i}[k+N-1|k]]'$$

and $P_{EM_i} = [P_{EM_i}[k] \ \dots \ P_{EM_i}[k+i] \ \dots \ P_{EM_i}[k+N-1]]'$ and R_i is the weight matrix (for example, an identity). Note that we set $N = 1$; hence, $P_{TC_i} = P_{TC_i}[k|k]$ and $P_{EM_i} = P_{EM_i}[k]$. Computing the products, we have

$$J = P_{TC_i}^T \cdot R_i \cdot P_{TC_i} - P_{TC_i}^T \cdot R_i \cdot P_{EM_i} - P_{EM_i}^T \cdot R_i \cdot P_{TC_i} + P_{EM_i}^T \cdot R_i \cdot P_{EM_i}.$$

Using the matrix rule $(A \cdot B)^T = B^T \cdot A^T$, we can rewrite the previous equation as

$$J = P_{TC_i}^T \cdot R_i \cdot P_{TC_i} - P_{EM_i}^T \cdot R_i \cdot^T P_{TC_i} - P_{EM_i}^T \cdot R_i \cdot P_{TC_i} + P_{EM_i}^T \cdot R_i \cdot P_{EM_i}.$$

The searched value is P_{TC_i}; hence, $s_i[k] = P_{TC_i}$. Then,

$$H_i = R_i \qquad g_i = -P_{EM_i}^T \cdot (R_i^T + R_i).$$

The term $P_{EM_i}^T \cdot R_i \cdot P_{EM_i}$ can be omitted since it is constant at each time step.

Now, we can manipulate the constraint inequality for obtaining M_i and b_i. Remembering that $N = 1$, (8) becomes

$$T_i[k+1|k] = C_i \cdot x_i[k+1] = C_i \cdot (A_i \cdot x_i[k] + B_i \cdot u_i[k]) \leq T_{MAX},$$

where $u = [P_{TC_i} \; T_{AMB} \; T_{NEIGH_i}]'$. Expliciting P_{TC_i}, we have

$$C_i \cdot B_{1_i} \cdot P_{TC_i} \leq T_{MAX} - C_i \cdot A_i \cdot x_i[k] - C_i \cdot B_{2_i} \cdot [T_{AMB} \;\; T_{NEIGH_i}]',$$

where $B = [B_{1_i} \; B_{2_i}]$ and B_{1_i} is the column of B_i related to the input P_{TC_i} and B_{2_i} is the completion of B_i. From the previous equation

$$M_i = C_i \cdot B_{1_i}$$

$$b_i = T_{MAX} - C_i \cdot A_i \cdot x_i[k] - C_i \cdot B_{2_i} \cdot [T_{AMB} \; T_{NEIGH_i}]'.$$

It is worth to note that we could also consider the case of a lower bound on the power ($P_{MIN} \leq P_{TC_i}$) writing

$$M_i = \begin{bmatrix} C_i \cdot B_{1_i} \\ -1 \end{bmatrix}$$

$$b_i = \begin{bmatrix} T_{MAX} - C_i \cdot A_i \cdot x_i[k] - C_i \cdot B_{2_i} \cdot [T_{AMB} \; T_{NEIGH_i}]' \\ -P_{MIN} \end{bmatrix}.$$

We do not take into account this power constraint in our simulation since it is reasonable to assume that the real chip is designed to dissipate a power higher than P_{MIN} without incurring in thermal issues.

Once the QP formulation is obtained, one of the two approaches can be used to find the optimal solution. The implicit approach uses a quadratic programming solver. It is worth to note that the matrix b_i depends on the current state $x_i[k]$ that is time variable. The same goes for $g_i[k]$ that depends on the requested power P_{EM_i} which is also time varying. Clearly, since the QP problem changes over time, the solving algorithm must be applied online at each time instant.

Although efficient QP solvers based on active-set methods and interior point methods are available, the computational overhead for finding the solution demands significant online computation effort. Assuming that the solution of the QP problem does not change much from the solution obtained at the previous iteration, we can reduce this effort by using an active set algorithm capable of finding the new solution starting the search from the previous one (hotstart). This algorithm is implemented in the open-source library qpOASES [46].

Another way to reduce computational burden is to use an explicit approach that solves the QP problem offline for all possible values of $x_i[k]$. The problem solved in this way is commonly called multiparametric QP (mp-QP). By treating $x_i[k]$ as a vector of parameters, the optimal solution is an explicit function of the state ($P_{TC_i}(x_i[k])$) with the property of being continuous piecewise affine [21]. In other words, it is possible to partition the state space into convex polyhedral regions, each one with its own optimal linear control law. At each time instant, $x_i[k]$ lies in one and only one of these regions. Similarly to a Look-Up Table (LUT), knowing the current state and by checking region boundaries it is possible to find the solution using the linear expression:

$$P_{TC_i}(x) = F_{r_i} \cdot x_i[k] + G_{r_i}, \tag{11}$$

where r is the region index and F_{r_i} and G_{r_i} are the corresponding gain matrices for each core i.

Even though on one hand the use of the explicit solution reduces the computational overhead, on the other hand it increases the memory usage for storing the data. Similarly to the overhead in the implicit solution, the number of regions increases with the problem complexity [22].

3.2.3 Observer Block

As previously introduced, the observer estimates the state values exploiting the current temperature of the system. We have used a classic Luenberger identity observer defined as

$$\hat{x}_i[k+1] = A_i \cdot \hat{x}_i[k] + B_i \cdot u_i[k] + K \cdot (T_i[k] - C_i \cdot \hat{x}_i[k]),$$

where $\hat{x}_i$ is the estimation of the state, A_i, B_i, C_i are the matrices obtained with the self-calibration routine, the same used by the MPC controller and u_i is the input vector containing P_{TC_i}, T_{AMB}, T_i, T_{NEIGH_i}. K is the observer gain matrix that is a degree of freedom for the designer. Its values are chosen to asymptotically stabilizing to zero the dynamic model of the estimation error $e[k] = \hat{x}_i[k] - x_i[k]$ characterized by the state matrix $(A_i - K_i \cdot C_i)$. To do that, we act on K_i to move the poles of the error model inside the unitary circle. In particular, we place the poles closer to the center than the poles of the model to have faster dynamics and thus a faster convergence of the error to zero. The only requirement to arbitrary move the poles is the observability of the pair (A_i,C_i) that is assured by the physics of the system, in fact each dynamic directly or indirectly affects the model output temperatures.

3.3 Self-Calibration Routine

The knowledge of the thermal model of the multicore die is a MPC prerequisite. In many practical cases, this model is not available or imprecise. We address model uncertainty by self-calibration: our thermal control framework directly learns the thermal model by monitoring the thermal response of the system after applying a set of training stimuli. This approach relies on system identification (SID) techniques. As shown in previous sections and in [47], complex prediction models complicate the control action and cause overhead. Thus, the implemented approach has two aims: capturing accurately the system thermal response and keeping small model size to represent the system (referred also as "plant" below) that would be characterized by infinite dimensional equations.

Indeed, our self-calibration routine is distributed: each core is associated with a self-identification agent that learns the local model. This approach perfectly fits our distributed control solution, since the regulator of each core directly exploits the identified local model for prediction. Second, it offers a low complexity solution to counteract the SID computational cost in large multicore systems. Indeed, for MIMO model, the SID complexity explodes with the number of inputs. Each agent (i) implements an AutoRegressive eXogenous (ARX) model [48], [49]

$$\begin{aligned} T[k] = \alpha_s \cdot T[k-1] + \cdots + \alpha_1 \cdot T[k-s] + \beta_{1,s} \cdot u_1[k-1] \\ + \cdots + \beta_{1,1} \cdot u_1[k-s] + \beta_{2,s} \cdot u_2[k-1] + \cdots + e[k], \end{aligned} \tag{12}$$

where T is the temperature of the core (the model output), s is the model order, $u_i(\cdot)$ are the model inputs (the dissipated power of the core P_{EM_i}, the ambient temperature T_{AMB}, and the temperatures of the neighbors units T_{NEIGH_i}), $e[k]$ is a stochastic white process with null expected value representing the model error, and α_m, $\beta_{m,j}$ are the identified parameters. As (12) shows, each model is a simple MISO model with a single output and multiple inputs $u_m(\cdot)$. The core power consumption can be estimated from the core operating point and from the current workload characteristic (see (1) in Section 2) or can be directly measured from power sensors present in recent MPSoC [28].

The self-calibration routine first forces a Pseudorandom Binary Sequence (PRBS) power input to each core, while probing the core temperature. Then, it derives the parameters α and β by solving a least square problem that minimizes the error $e[k]$. To take into account both the slow and the fast dynamics highlighted in Section 2, we use a second-order ARX model ($s = 2$) for each core and then we convert it in the state-space form $(A_{ARX_i}, B_{ARX_i}, C_{ARX_i})$.

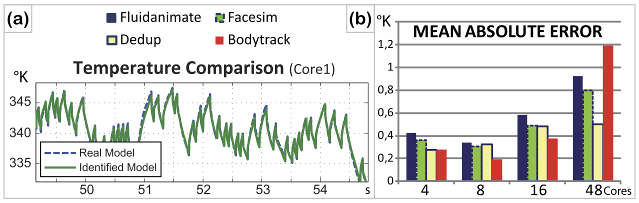


Fig. 6. Self-calibration routine results.

Unfortunately, the identified models states do not have a physical meaning. To match the core temperature with the first state of each model, we apply a change of coordinate transformation to obtain a matrix $C = [I_s \mid 0_s]$ where I_s is the s-dimensional identity matrix. We achieve that by representing our system in the equivalent observer canonical form. We use the observability matrix $O = [C_{ARX_i};\ C_{ARX_i} \cdot A_{ARX_i}]$ as linear transformation to change the coordinates of the $(A_{ARX_i}, B_{ARX_i}, C_{ARX_i})$ model as shown below:

$$\begin{cases} A_i = O^{-1} \cdot A_{ARX_i} \cdot O \\ B_i = O^{-1} \cdot B_{ARX_i} \\ C_i = C_{ARX_i} \cdot O. \end{cases} \tag{13}$$

Since each thermal model is a second-order model and each element of A_i, B_i, C_i can be expressed as the composition of a finite algebraical operation of the element of A_{ARX_i}, B_{ARX_i}, C_{ARX_i}, the above computation is negligible.

We test the performance of our technique using a plant and a high dimensional and accurate model developed in Matlab using a finite elements approach [50]. In Fig. 6a, we have shown the comparison between the plant and the model temperature of core 1 obtained applying PRBS signals different from those used for the self-calibration routine. In Fig. 6b, instead we validate our technique on plants with increasing number of cores. First, we have applied PRBSs on each core and then we have computed the model. Finally, we have measured the mean absolute error between the temperature of the system and the temperature of the model, both running Parsec2.1 benchmarks traces. The resulting errors are lower than 1 K on average.

The showed results are obtained running simulations on Matlab/Simulink environment. In a real system, we expect to run the self-calibration routine during start up phase and each time the model behavior differs from plant one.

3.4 Implementation

In this section, we describe the pseudocode of our implementation. First, during system initialization, the self-calibration routine described in Section 3.3 is executed for each core to gather the local thermal model. Second, as shown in Section 3.2.2, with the extracted local thermal model we update the weight of each local QP problem. Third, at each sample, we execute the energy mapper and thermal controller routines. Subsequently at each controller step we apply the optimal frequency to the controlled core.

The code below shows the list of operations executed in parallel by each core to gather the local thermal model, adapt the controller parameters, and control the system performance.

Pseudo Code

```
SYSTEM INITIALIZZATION PHASE:                                        1
  apply a PRBS task sequence;                                        2
     get $P_i, T_i, T_{NEIGH_i}$;                                    3
     obtain the $A_{ARX_i}, B_{ARX_i}, C_{ARX_i}$ with the $ARX$;    4
     change the state coordinate $\rightarrow \{A_i, B_i, C_i\}$;    5
CONTROLLER ROUTINE                                                   6
  Initialize $H_i$, $M_i$, $\hat{x}_i(0)$;                           7
  FOR EACH SAMPLE                                                    8
          get previous $f_{TC_i}[k-1]$ and $CPI_i[k-1]$;             9
          compute the optimal $f_{EM_i}[k]$;                        10
          convert to $P_{EM_i}$ using ((1));                        11
          update $g_i$ and $b_i$;                                   12
          solve QP and find $P_{TC_i}[k]$ with hotstart;            13
          compute $f_{TC_i}[k]$;                                    14
          estimate of $\hat{x}_i[k]$;                               15
  END_FOR                                                           16
```

More in detail:

- Line 1: during the system initialization phase, we execute in parallel in each core the self-calibration routine;
- Line 2: we apply a pseudorandom task sequence to each core and we collect the core power and local neighborhood temperature traces;
- Line 4: we execute the ARX optimization in each core to obtain the $A_{ARX_i}, B_{ARX_i}, C_{ARX_i}$ state-space model;
- Line 5: we compute the observability matrix and we change the state-space coordinates. The measured output vector gains the physical meaning of core temperature vector;
- Line 7: we first define the constant matrices used in the QP problem (H_i, M_i) and assign the initial state, $\hat{x}_i(0)$, to the model;
- Line 8: at each time step, the loop from lines 8 to 16 is repeated;
- Line 9: the Energy Mapper read the previous step core frequency $f_{TC_i}[k-1]$ and $CPI_i[k-1]$, and it estimates the future workload requirement $CPI_i[k]$ and compute optimal target frequency $f_{EM_i}[k]$;
- Line 11: the Thermal Controller receives the target frequency and the workload from the EM and then it converts them into the target power using the nonlinear function (1);
- Line 12: vector g_i, dependent on $P_{EM_i}[k]$, and vector b_i, dependent on $\hat{x}_i[k]$, $T_{AMB}[k]$ and $T_{NEIGH_i}[k]$, are updated.
- Line 13: starting from the previous optimal solution $P_{TC_i}[k-1]$ the solver finds the optimal $P_{TC_i}[k]$, solution of the QP problem.
- Line 14: P_{TC_i} and workload are used to find f_{TC_i} inverting the function (1) with the Brent's algorithm (Section 3.2.1).
- Line 15: the observer estimates the state $\hat{x}[k]$ knowing the $f_{TC_i}[k]$ and $T_i[k]$. This state will be used as initial state by the MPC controller to estimate the future temperature of the core.

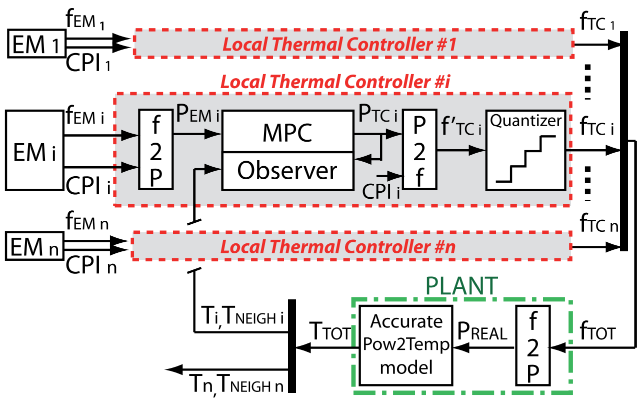


Fig. 7. Simulator block diagram.

The explicit MPC implementation is similar to the one described above with the only difference in lines 12-13. Indeed, the optimal P_{TC_i} is obtained with (11). As we will discuss in Section 4 both implementations, thank to our distributed strategy and the hotstart QP solver (in the implicit implementation), have low overhead and are suitable to be executed with a sample rate of 1-10 ms.

4 Experimental Results

In this section, we illustrate the results obtained using the distributed approach. These results are obtained by means of experimental tests performed with two different types of simulator. The first has been realized on Matlab/Simulink [52] environment to preliminary test our control solution on application traces. Fig. 7 shows the simulation infrastructure. As we have seen, each local thermal controller takes as input the CPI_i and the frequency requirement, f_{EM_i}, coming from the EM_i. Internally, these values are translated into a power target for the MPC (P_{EM}). The MPC uses it in combination with the observer and the temperature sensors reading (T_i, T_{NEIGH_i}), coming from the plant, to compute the optimal power values P_{TC_i}. Then, it computes the optimal frequency to be applied to the plant f_{TC_i}. To have more realistic simulation and take into account actuator nonidealities, we quantize the f_{TC_i} into 100 MHz steps.

The plant is a model of the real system. It takes as input the frequencies and returns the temperatures of all cores. Similarly to what has been said in Section 2, the plant is split in a nonlinear part which uses a vector formulation of (1) for translating the frequencies into powers and a linear part which models the chip power to temperature relation, as (2) does for the single core. The only requirement to design this linear model is the accuracy with respect to the real system. Ideally, it should have an infinite dimension to capture all possible dynamics. Since this is practically infeasible, we have built a model with very high dimension (more than 10 times the dimension of the identified model), exploiting the finite element approach described in [50].

In Sections 4.1 and 4.2 are shown the experimental results attained using the simulator described above. The first test show a set of comparisons to clarify our design choices, while the second highlights the scalability advantages of our distributed solution respect to the state-of-the-art centralized ones. In Section 4.3, we use a full-system virtual platform to gauge with high accuracy the complex

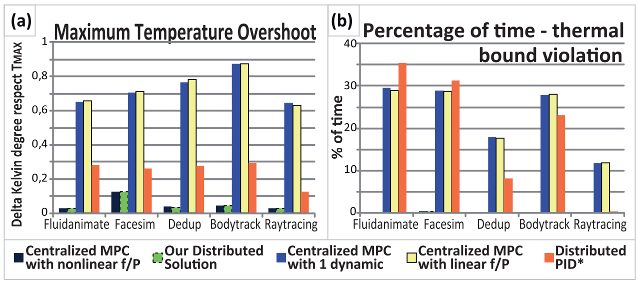


Fig. 8. Design choices results.

interdependencies between control action and workload, which is lost when using trace-based workload models.

4.1 Design Choices Motivation

We have performed a set of tests to analyze the quality of our controller with respect to the design choices we have taken. Before describing the tests, we define the state-of-the-art centralized solution that will be our yardstick since it guarantees the maximum achievable thermal controller performance, reflecting in minimal QoS degradation due to thermal constraining. We can obtain the centralized solution from the presented distributed one adapting the model and the optimization problem. The model is a MIMO model that takes as input all P_{EM_i}, T_{AMB} and does not need for any T_{NEIGH_i}, while the optimization problem has the same cost function and constraint inequality except for P_{TC}, P_{EM}, and T that are vectors containing all core powers and temperatures. The first test measures the importance of using a nonlinear model inside the controller to predict the future temperatures. We compare the centralized solution with a similar one where a linear function has been used for $f2P$ and $P2f$ conversions. As in [8], we use the best first degree polynomial fitting the nonlinear function.

The second test, instead, shows the importance of having an accurate model with the right number of dynamics for predictions. We compare the centralized solution with two dynamics with the same solution, but with a simpler first-order model as in [8].

Note that the above tests show the relevance of these approximations in state-of-the-art solutions that are centralized. An important result concerning our distributed solution is obtained in the third test, where we compare it with the centralized optimal one.

Finally, a comparison with a distributed PID-based solution is performed to highlight the goodness of our solution.

The experimental results in Figs. 8a and 8b refer to a four cores chip where different benchmarks are run. Notice that the number of cores does not impact on the results of the comparison tests we perform, increasing the cores only emphasizes the results attained. The metrics used to analyze the controller performance are the maximum temperature overshoot with respect to the constraint and the percentage of time the temperatures violate the constraint (we consider the limit violated when temperature exceeds 330.1 K).

From the first test, the MPC with linear f2P function shows globally lower controller performance. It has bigger maximum overshoot, and the time percentage the constraint is violated is significantly higher. This is because the linear function transforms the controller output power, P_{TC}, into an input frequency for the plant model which is higher or lower than the frequency really derived by the controller. Notice that performance is influenced by the previous results since the frequencies are maintained closer to the input target frequencies because the temperature constraint is violated more than the other cases. To further stress our approach we performed a sensitivity test respect to f2P and P2f accuracy. Indeed, as we show in the supplemental online section, with a 20 percent of induced error in the power model parameters the controller performance slightly degrades, with maximum overshots below $0.2\ K$ and the time percentage of bound violation below 10 percent.

The test on model order shows as we expected a worsening of the controller performance. The first-order model we use considers only the slower dynamic (see Section 2), that is the dominant. The absence of faster dynamics causes oscillations in the controlled temperature that causes a large increase in the percentage of time the constraint is violated. The controller sees that the model behavior is characterized by slow dynamics so in proximity of the temperature limit it gives a strong control action to rapidly decrease the temperature. Differently, the plant, having both slow and fast dynamics, responds with larger temperature decreasing. The controller senses this unexpected drop in the temperature and, again, it reacts with a control action larger than what needed. This gives rise to a limit-cycle oscillating behavior which is far from ideal.

The third test shows that the distributed and centralized solutions have comparable performance. Even though performance obtained by the centralized solution represents an upper bound for the distributed one, we attain a close approximation of the optimal control actions computed by the centralized controller. Consequently, the results of the first two tests hold also for the distributed solution.

Finally, the fourth test compares the performance of our distributed solution with a PID controller. It is relevant to remark that the PID solution is power driven and not frequency driven. Moreover, the solution does not include the quantization and has been discretized with a finer sampling time (0.5 ms) to take advantage of its internal lower complexity. Despite these “advantages” for PID, our distributed solution performs better. Major details on PID design are shown in the supplemental online section.

4.2 Scalability and Complexity Tests

In previous section, we demonstrate that our distributed solution performs as the centralized one in terms of controller quality. In this section, we will instead show its main benefits in terms of computational complexity and scalability.

Indeed, we first compare the distributed and the centralized solutions for increasing number of cores. To evaluate the complexity, we use as metrics both the execution time, for the implicit formulation, and the number of regions for explicit one. As a matter of fact, Zanini et al. [22] show that the number of operations depends logarithmically on the number of regions. Instead for the implicit solution, we provide the mean computational time necessary to solve the QP problem at each iteration. This time has been obtained running a C code version of the control algorithm on a 2.4 GHz dual-core processor. Fig. 9a shows how the complexity of centralized MPC solutions increases with the number of cores. The time spent to solve the QP problem and the number of regions exponentially increases with the number

a) *Centralized MPC Complexity*

	MPC_explicit	MPC_implicit
4	81	7,70
8	6561	9,00
16	OUT	24,20
48	OUT	85,50
	# regions	time (us)

b) *Distributed MPC Complexity (single core)*

f2P	0,061	time (us)
Observer	0,743	time (us)
MPC (Impl)	4,690	time (us)
MPC (Expl)	2	# regions
P2f	1,188	time (us)

Fig. 9. Scalability and complexity reduction results.

of cores in the worst case. In particular, as asserted in [21], the regions number increases with the number of states, decision variables, and constraints. For a chip with 16 or 48 cores, we were not been able to compute them for memory limitations.

Fig. 9b instead shows the complexity of our distributed solution: whereas the centralized solution grows exponentially, the distributed one globally increases linearly. Moreover, the complexity of a single controller node remains constant regardless the number of cores. Indeed, each one has only two regions and spends on average only $4,69\ \mu s$ to solve the QP problem. The same figure also shows the execution time for the frequency-to-power conversion and the observer estimation. Notice that their impact on the global execution time is negligible ($<2\ \mu s$).

In addition, we need to consider that our distributed controller is naturally parallel (i.e., one local controller running on each core), while parallelization of the centralized controller is far from obvious. As consequence of our distributed implementation, each thermal controller can be stopped autonomously without impacting on other controllers performance. This avoids periodically waking up of idle cores to execute the controller routine. Moreover, the small power consumption of cores, when power gated, ensures that no thermal emergencies can occur on their surface. Consequently, they do not need to be thermally controlled. As the core is waken up, the controller will pay a temporally QoS loss due to the partially valid initial state vector. Properties in the Luenberger observer ensure this error fast converges to zero. This is not the case of centralized MPC where the core in which the controller is running always need to be periodically waken up since other cores might be under thermal emergencies. This introduces extra energy penalties in centralized solution applicability. To conclude, Fig. 10 shows the spatial performance of our distributed controller on a 48 cores chip in two different situations. The two figures on the top (a,b) refer to the case where the 16 central cores are stressed, whereas the two figures on the bottom (c,d) refer to the case where half of the cores are stressed. In Figs. 10a and 10c, each vertex represents the temperature measured by the sensor of the respective core. Figs. 10b and 10d instead show the frequency of each core. Initially, all frequencies are equal to 3,000 MHz and all cores are in idle. Then, to the 16 central cores in a,b and to the 24 left cores in c,d is allocated a power virus. When the temperature reaches the threshold of $330\ K$, the thermal controller reduces the frequency of the hottest cores enough to remain below the limit. Note that in both cases the stressed cores on the boundary are subjected to a lower frequency reduction since a significant part of the heat is dissipated through adjacent cold cores.

4.3 Full-System Simulation

As previously introduced, we implemented the proposed solution in the virtual platform presented in [37]. This virtual

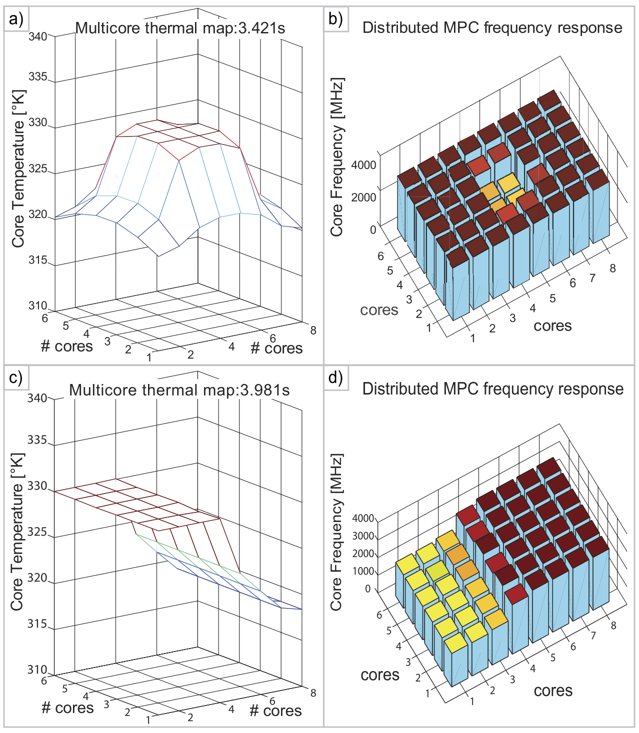


Fig. 10. Forty-eight cores chip simulation with distributed control regulation.

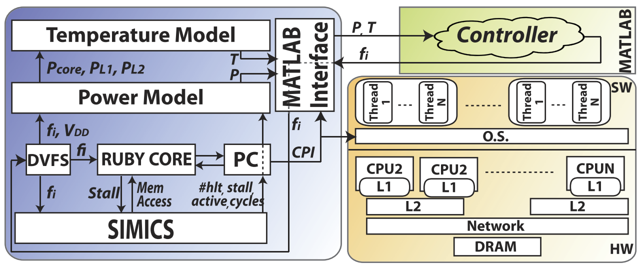


Fig. 11. Virtual platform overview.

platform emulates a general-purpose multicore running in a full system. It provides a flexible and effective tool to support the design space exploration of power, thermal, and reliability control-theory-based close-loop resource management solution. It relies on a x86 ISA functional simulator, namely SIMICS [53], combined with GEMS [54], a complex memory hierarchy timing-accurate model.

We integrated in GEMS the support to per-core dynamic voltage and frequency scaling and the emulation of per-core performance counters and temperature sensors. As shown in the Fig. 11, temperature distribution and power dissipation are evaluated by integrating in the virtual platform a power and a temperature simulator. Both the power and temperature simulators are calibrated on a real multicore.[7] Finally, the chosen virtual platform allows a Matlab/Simulink description of the controller to execute natively as a new component of the virtual platform and directly drive the performance knobs of the emulated system.

We exploited this infrastructure to implement our thermal controller solution as Matlab routines and fully integrating it in the virtual platform by using the Matlab interface. For each core (i) in the emulated system we execute, with a time step of 1 ms, the Energy Mapper (EM_i) and Thermal Controller (TC_i) routines. The Thermal Controller routine embeds, as presented in Section 3.2, the explicit MPC implementation and estimates full state with the state observer. Even if the controller routines are not executed directly on the target multicore, the complexity analysis in Section 4.2 demonstrates that the distributed solution has negligible runtime. Thus, the perturbation due to its computations to the program execution flow can be neglected.

7. Xeon X7350.

We test our proposed solution on the virtual platform by using the target architecture in [37] and running on each core different Parsec2.1 [51] benchmarks workloads. All benchmarks have been executed with a number of tasks equal to the number of cores of the target architecture, with the input set "simsmall," and constrain temperature $(T_{MAX} = 330\ K)$.[8] We run each Parsec benchmark under four possible configurations: Original, only Energy Mapper, Centralized Thermal Controller (Centr TC), and our Distributed Thermal Controller (Distr TC). In this set of tests, we use a QoS Loss metric that quantifies the controller quality-of-service degradation due to thermal constraint. We decide to compute it as the mean-squared error between the energy mapper frequency target (f_{EM}) and the one applied to the system by the controller (f_{TC}). We relativized it against the centralized controller one.

Fig. 12a shows the performance of the EM alone, while allowing a performance penalty of $T_{\%,i}[k] = 5\%$. We can notice that it is able to maintain the performance overhead under the selected threshold while achieving a significant power and energy saving.

For all the configurations, we extract the performance metrics discussed in Section 4.1. Fig. 12 shows the results collected. First, we can notice that the proposed distributed solution performs as well as the centralized one. Fig. 12b shows the maximum overshoot in Kelvin degrees above the safety thermal threshold (T_{MAX}), whereas Fig. 12c shows that both solutions are capable of drastically reducing the portion of time in which each core runs out of the thermal bound. Looking at the QoS Loss performance figure (Fig. 12d), we notice that our proposed solution performs at the same level of the centralized one, with a degradation less than 3 percent. Finally, in more symmetrical workloads,[9] such as `swaptions`, `fluidanimate`, `canneal`, we noticed that the average frequency applied to the external cores (#1, #4) is kept lower (up to -14 percent) than the internal cores. This is a sign that the MPC controller is able to optimize the core frequency locally, taking advantage of the difference between the local thermal models. Indeed, the external thermal models have less thermal dissipation headroom since thermal model considers the chip lateral boundary adiabatic [50].

5 Conclusion

We have presented a novel fully distributed, energy-aware, MPC-based thermal capping controller for modern multicore platforms. It works coupled with a model self-calibration

8. Used thermal model is calibrated on a device with high performance thermal dissipation dynamics, indeed to stress our thermal controller we are forced to use a lower temperature constraint.

9. The parallel benchmark executes the same code on all the processors.

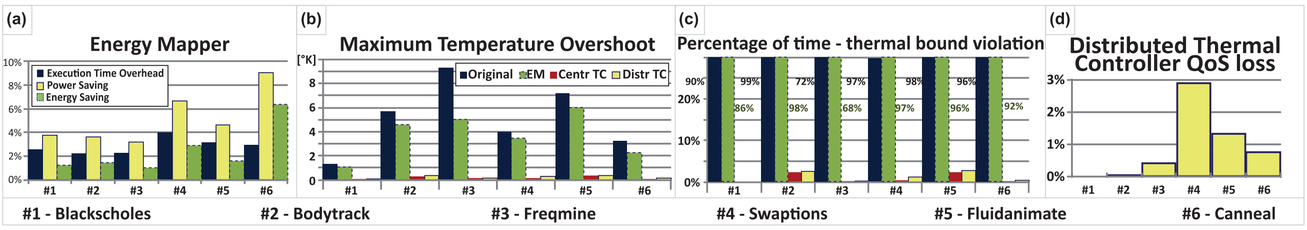


Fig. 12. Virtual platform test results.

routine, that allows the controller to automatically recognize the thermal properties of the underlying platform. It exploits a fully distributed architecture, that fits very well the multicore parallelism and distributed nature. We show that this approach performs similarly to the state-of-the-art centralized Thermal Model Predictive Controllers, but with a significantly lower computational cost. Indeed, the global complexity cost of our solution scales linearly with the number of cores and it is fully parallel, whereas the centralized one scales exponentially and its parallelization is not trivial. We tested our solution in a real use case scenario by running it in a complete virtual platform. The results show that our controller is capable to satisfy temperature constraints with high precision, while minimizing system energy.

Acknowledgments

This work was supported, in parts, by Intel Corp., Intel Labs Braunschweig, the EU FP7 ERC Project MULTITHERMAN (GA n. 291125) and the EU FP7 Project Therminator (GA n. 248603).

References

[1] "Make IT Green," http://www.greenpeace.org/international/Global/international/planet-2/report/2010/3/make-it-green-cloud-computing.pdf, 2012.

[2] "SMART 2020: Enabling the Low Carbon Economy in the Information Age," http://www.smart2020.org/_assets/files/01_Smart2020ReportSummary.pdf, 2008.

[3] M. Monchiero, R. Canal, and A. Gonzalez, "Power/Performance/Thermal Design-Space Exploration for Multicore Architectures," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 5, pp. 666-681, May 2008.

[4] Active Power "Data Center Thermal Runaway. A Review of Cooling Challenges in High Density Mission Critical Environments," http://www.activepower.com, 2007.

[5] J. Moras, "HPC Efficiency and Scalability through Best Practices: Lessons Learned," *Proc. HPC Advisory European Council Workshop,* http://www.hpcadvisorycouncil.com, May 2010.

[6] G. Contreras and M. Martonosi, "Techniques for Real-System Characterization of Java Virtual Machine Energy and Power Behavior," *Proc. IEEE Int'l Symp. Workload Characterization (IISWC),* Oct. 2006.

[7] R. Yavatkar and M. Tirumala, "Platform Wide Innovations to Overcome Thermal Challenges," *Microelectronics J.,* vol. 39, no. 7, pp. 930-941, 2008.

[8] X. Wang, K. Ma, and Y. Wang, "Adaptive Power Control with Online Model Estimation for Chip Multiprocessors," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 10, pp. 1681-1696, Oct. 2011.

[9] G. Dhiman and T.S. Rosing, "Dynamic Voltage Frequency Scaling for Multi-Tasking Systems Using Online Learning," *Proc. Int'l Symp. Low Power Electronics and Design (ISLPED),* Aug. 2007.

[10] G. Keramidas, V. Spiliopoulos, and S. Kaxiras, "Interval-Based Models for Run-Time Dvfs Orchestration in Superscalar Processors," *Proc. Seventh ACM Int'l Conf. Computing Frontiers (CF '10),* pp. 287-296, 2010.

[11] J. Kong, S.W. Chung, and K. Skadron, "Recent Thermal Management Techniques for Microprocessors," *ACM Computing Survey,* vol. 44, article 13, 2012.

[12] P. Chaparro, J. Gonzalez, G. Magklis, Q. Cai, and A. Gonzalez, "Understanding the Thermal Implications of Multi-Core Architectures," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 8, pp. 1055-1065, Aug. 2007.

[13] M.M. Sabry, D. Atienza, and A.K. Coskun, "Thermal Analysis and Active Cooling Management for 3D MPSoCs," *Proc. Int'l Symp. Circuits and Systems (ISCAS),* 2011.

[14] K. Kang, J. Kim, S. Yoo, and C.M. Kyung, "Runtime Power Management of 3-D Multi-Core Architectures under Peak Power and Temperature Constraints," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 6, pp. 905-918, June 2011.

[15] X. Zhou, J. Yang, Y. Xu, Y. Zhang, and J. Zhao, "Thermal-Aware Task Scheduling for 3D Multicore Processors," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 1, pp. 60-71, Jan. 2010.

[16] A.K. Coskun, J.L. Ayala, D. Atienza, T.S. Rosing, and Y. Leblebici, "Dynamic Thermal Management in 3D Multicore Architectures," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE),* pp. 1410-1415, Apr. 2009.

[17] K. Skadron, T. Abdelzaher, and M.R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," technical report, UMI Order Number: CS-2001-27., Univ. of Virginia, 2001.

[18] M. Kadin, S. Reda, and A. Uht, "Central versus Distributed Dynamic Thermal Management for Multi-Core Processors: Which One Is Better?" *Proc. 19th ACM Great Lakes Symp. VLSI (GLSVLSI),* pp. 137-140, 2009.

[19] Z. Wang, X. Zhu, C. McCarthy, P. Ranganathan, and V. Talwar, "Feedback Control Algorithms for Power Management of Servers," *Proc. Int'l Workshop Feedback Control Implementation and Design in Computing Systems and Networks (FeBid),* June 2008.

[20] E.F. Camacho and C. Bordons, *Model Predictive Control.* Springer, 1999.

[21] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica,* vol. 38, pp. 3-20, 2002.

[22] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore Thermal Management with Model Predictive Control," *Proc. European Conf. Circuit Theory and Design (ECCTD),* vol. 1, 2009.

[23] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Thermal-Aware System-Level Modeling and Management for Multi-Processor Systems-On-Chip," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS),* pp. 2481-2484, 2011.

[24] R. de la Guardia and W.M. Beltman, "Advanced Acoustic Management for PCs," *InterNoise,* 2006.

[25] R. Cochran and S. Reda, "Consistent Runtime Thermal Prediction and Control through Workload Phase Detection," *Proc. 47th Design Automation Conf. (DAC),* pp. 62-67, 2010.

[26] T.J.A. Eguia, S.X.-D. Tan, R. Shen, E.H. Pacheco, and M. Tirumala, "General Behavioral Thermal Modeling and Characterization for Multi-Core Microprocessor Design," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE),* 2010.

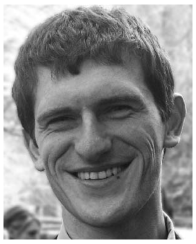
[27] P. Kumar and D. Atienza, "Runtime Adaptable On-Chip Thermal Triggers," *Proc. Asia and South Pacific Design Automation Conf. (ASPDAC),* pp. 255-260, 2011.

[28] J. Howard et al., "A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS," *Proc. Int'l Solid-State Circuits Conf. (ISSCC),* 2010.

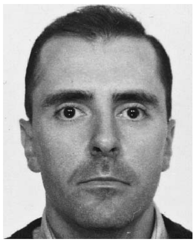
[29] F. Zanini, C.N. Jones, D. Atienza, and G. De Micheli, "Multicore Thermal Management Using Approximate Explicit Model Predictive Control," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS),* pp. 3321-3324, May/June 2010.

[30] A.K. Coskun, T.S. Rosing, and K.C. Gross, "Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 28, no. 10, pp. 1503-1516, Oct. 2009.
[31] A. Mutapcic and S. Boyd, "Processor Speed Control with Thermal Constraints," *IEEE Trans. Circuits and Systems I,* vol. 56, no. 9, pp. 1994-2008, Sept. 2009.
[32] E. Camponogara, D. Jia, H. Krogh, and S. Talukdar, "Distributed Model Predictive Control," *IEEE Control Systems,* vol. 22, no. 1, pp. 44-52, Feb. 2002.
[33] R. Scattolini, "Architectures for Distributed and Hierarchical Model Predictive Control," *J. Process Control,* vol. 19, no. 5, pp. 723-731, May 2009.
[34] S. Reda, R. Cochran, and A.N. Nowroz, "Improved Thermal Tracking for Processors Using Hard and Soft Sensor Allocation Techniques," *IEEE Trans. Computers,* vol. 60, no. 6, pp. 841-851, June 2011.
[35] S. Sharifi and T.S. Rosing, "Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 29, no. 10, pp. 1586-1599, Oct. 2010.
[36] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "A Distributed and Self-Calibrating Model-Predictive Controller for Energy and Thermal Management of High-Performance Multicores," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE),* 2011.
[37] A. Bartolini, M. Cacciari, A. Tilli, L. Benini, and M. Gries, "A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-Performance Multicores," *Proc. 20th Symp. Great Lakes Symp. VLSI (GLSVLSI),* 2010.
[38] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and Responsive Power Models for Multicore Processors Using Performance Counters," *Proc. ACM Int'l Conf. Supercomputing (ICS '10),* pp. 147-158, 2010.
[39] K. Singh, M. Bhadauria, and S.A. McKee, "Real Time Power Estimation and Thread Scheduling via Performance Counters," *SIGARCH Computer Architecture News,* vol. 37, no. 2, pp. 46-55, July 2009.
[40] X. Chen, C. Xu, R.P. Dick, and Z. Morley Mao, "Performance and Power Modeling in a Multi-Programmed Multi-Core Environment," *Proc. 47th Design Automation Conf. (DAC '10),* pp. 813-818, 2012.
[41] V. Hanumaiah, S. Vrudhula, and K.S. Chatha, "Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-Core Processors," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems (CAD),* vol. 30, no. 11, pp. 1677-1690, Nov. 2011.
[42] N. Sakran, "The Implementation of the 65nm Dual-Core 64b Merom Processor," *Proc. IEEE Int'l Solid-State Circuits Conf. (ISSCC),* 2007.
[43] H. Wei et al., "Accurate, Pre-RTL Temperature-Aware Design Using a Parameterized, Geometric Thermal Model," *IEEE Trans. Computers,* vol. 57, no. 9, pp. 1277-1288, Sept. 2008.
[44] W. Huang, K. Skadron, S. Gurumurthi, R.J. Ribando, and M.R. Stan, "Differentiating the Roles of IR Measurement and Simulation for Power and Temperature-Aware Design," *Proc. Int'l Symp. Performance Analysis of Systems and Software (ISPASS),* 2009.
[45] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing,* second ed. Cambridge Univ. Press, 1992.
[46] qpOASES Homepage, http://www.qpOASES.org/, 2011.
[47] F. Zanini, D. Atienza, G. De Micheli, and S.P. Boyd, "Online Convex Optimization-Based Algorithm for Thermal Management of MPSoCs," *Proc. 20th Symp. Great Lakes Symp. VLSI (GLSVLSI),* 2010.
[48] L. Ljung, *System Identification - Theory for the User,* second ed. PTR Prentice Hall, 1999.
[49] R. Guidorzi, *Multivariable System Identification.* Bononia Univ. Press, 2003.
[50] G. Paci, M. Morari, V. Dua, and E.N. Pistikopoulos, "Exploring Temperature-Aware Design in Low-Power MPSoCs," *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE),* vol. 1, pp. 1-6, 2006.
[51] C. Bienia, S. Kumar, J.P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," *Proc. 17th Int'l Conf. Parallel Architectures and Compilation Techniques (PACT),* 2008.
[52] The MathWorks. MATLAB & Simulink, http://www.mathworks.com/, 2012.
[53] Virtutech. Virtutech Simics, http://www.virtutech.com/, 2012.
[54] M.M.K. Martin et al., "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) toolset," *SIGARCH Computer Architecture News,* vol. 33, no. 4, pp. 92-99, 2005.

Andrea Bartolini received the engineering degree in electronic engineering from the University of Bologna, Italy, in 2007. He received the PhD degree in electronic engineering from the University of Bologna, Italy, in 2011. He is currently a postdoc researcher in the Department of Electrical Engineering and Computer Science (DEIS) at the University of Bologna. His research interests concern dynamic resource management of embedded systems and multi-core systems with special emphasis on software level thermal and power aware techniques. His research interest includes also simulation and operating system design for many-core devices.



Matteo Cacciari received the engineering degree in system engineering from the University of Bologna, Italy, in 2009. He is currently working toward the PhD degree in the Department of Electronic and Computer Science (DEIS) at the University of Bologna. His research interests include, constrained predictive control, convex optimization, thermal control of MPSoC, and Software-in-the-loop simulation.



Andrea Tilli received the DrIng degree in electronic engineering and the PhD degree in automatic control from the University of Bologna, Italy, in 1996 and 2000, respectively. Since 1997, he has been with the Department of Electronics, Computer, and System Science (DEIS), University of Bologna, where, in 2001, he became a research associate. His current research interests include applied nonlinear control techniques, thermal control for multicore SoC, power electronics equipments, active power filters, wind turbines, electric drives and automotive control systems. He was the recipient of a research grant from DEIS on modeling and control of complex electromechanical systems in 2000.

Luca Benini received the PhD degree in electrical engineering from Stanford University in 1997. He is a full professor at the University of Bologna. He also holds a visiting faculty position at the Ecole Polytechnique Federale de Lausanne (EPFL). His research interests are in the design of system-on-chip platforms for embedded applications. He is also active in the area of energy-efficient smart sensors and sensor networks for biomedical and ambient intelligence applications. He has published more than 500 papers in peer-reviewed international journals and conferences, four books and several book chapters. He has been general chair and program chair of the Design Automation and Test in Europe Conference. He has been a member of the technical program committee and organizing committee of several conferences, including the Design Automation Conference, International Symposium on Low Power Design, the Symposium on Hardware-Software Codesign. He is an associate editor of several international journals, including the *IEEE Transactions on Computer Aided Design of Circuits and Systems* and the *ACM Transactions on Embedded Computing Systems*. He is a fellow of the IEEE and a member of the steering board of the ARTEMISIA European Association on Advanced Research and Technology for Embedded Intelligence and Systems.