Online Thermal Control Methods for Multiprocessor Systems

FRANCESCO ZANINI, DAVID ATIENZA, and COLIN N. JONES, Ecole Polytechnique Fédérale de Lausanne (EPFL)
LUCA BENINI, Universitá di Bologna
GIOVANNI DE MICHELI, Ecole Polytechnique Fédérale de Lausanne (EPFL)

With technological advances, the number of cores integrated on a chip is increasing. This in turn is leading to thermal constraints and thermal design challenges. Temperature gradients and hotspots not only affect the performance of the system but also lead to unreliable circuit operation and affect the lifetime of the chip. Meeting temperature constraints and reducing hotspots are critical for achieving reliable and efficient operation of complex multi-core systems.

In this article, we analyze the use of four of the most promising families of online control techniques for thermal management of multiprocessors system-on-chip (MPSoC). In particular, in our exploration, we aim at achieving an online smooth thermal control action that minimizes the performance loss as well as the computational and hardware overhead of embedding a thermal management system inside the MPSoC. The definition of the optimization problem to tackle in this work considers the thermal profile of the system, its evolution over time, and current time-varying workload requirements. Thus, this problem is formulated as a finite-horizon optimal control problem, and we analyze the control features of different online thermal control approaches. In addition, we implemented the policies on an MPSoC hardware simulation platform and performed experiments on a cycle-accurate model of the eight-core Niagara multi-core architecture using benchmarks ranging from Web-accessing to playing multimedia. Results show different trade-offs among the analyzed techniques regarding the thermal profile, the frequency setting, the power consumption, and the implementation complexity.

Categories and Subject Descriptors: J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design (CAD)*

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: MPSoC, thermal management, DVFS, online, MPC, convex optimization, hot spots

ACM Reference Format:
Zanini, F., Atienza, D., Jones, C. N., Benini, L., and De Micheli, G. 2012. Online thermal control methods for multiprocessor systems. ACM Trans. Des. Autom. Electron. Syst. 18, 1, Article 6 (December 2012), 26 pages. DOI = 10.1145/2390191.2390197 http://doi.acm.org/10.1145/2390191.2390197

6

This research is supported in part by ERC Senior Grant 246810 and by the PRO3D EU FP7-ICT-248776 project.
Authors' addresses: F. Zanini and G. De Micheli, EPFL-IC-ISIM-LSI, Station 14, CH-1015 Lausanne, Switzerland; D. Atienza, EPFL-STI-IEL-ESL, Station 11, CH-1015 Lausanne, Switzerland; C. N. Jones, Automatic Control Laboratory (LA), EPFL-STI-IGM-LA Station 9, CH-1015 Lausanne, Switzerland; email: {francesco.zanini, goivanni.demicheli, david.atienza, colin.jones}@epfl.ch; L. Benini, Dipertimento di Elettronica, Informatica e Sistemistica (DEIS), University of Bologna, Viale Risorgimento 2, IT-40136 Bologna, Italy; email: luca.benini@dunibo.it.

DOI 10.1145/2390191.2390197 http://doi.acm.org/10.1145/2390191.2390197

1. INTRODUCTION

With technology scaling, the number of transistors available on a chip and their speed of operation is increasing rapidly. Today, several commercial multi-core architectures ranging from few cores to several tens of cores are available, such as Oracle's Niagara [Kongetira et al. 2005]. Power and thermal management are critical challenges for multi-core systems [Borkar 1999]. Temperature gradients and hotspots not only affect the performance of the system but also lead to unreliable circuit operation and reduce the lifetime of the chip [Skadron et al. 2004].

Temperature gradients are a concern not only in space but also in time. Frequent abrupt changes in working frequencies and voltages produce thermal cycles that raise the failure rate of the system [Haase et al. 2006]. The effect of thermal cycling on the reliability of a chip can be modeled by the Coffin-Manson relation, which links in an exponential way the number of cycles to failure to the magnitude of thermal cycling [JEDEC 2003]. In addition, abrupt power-mode transitions in voltage and frequency scaling waste additional power [Jung and Pedram 2008].

In recent years, thermal management techniques have received a lot of attention. Many state-of-the-art thermal control policies manage power consumption via *dynamic frequency and voltage scaling* (DVFS) [Mukherjee and Memik 2006]. The structure of state-of-the-art DVFS-based thermal management systems is reported in the diagram of Figure 1.

The thermal management policy regulator monitors the *multiprocessors system-on-chip* (MPSoC). Without loss of generality, we assume this MPSOC to be partitioned into p islands (or subsystems), each with independent frequency and voltage settings. For our purposes, we consider frequencies as inputs to the system, since we are abstracting away the computation.

The regulator sets working frequencies according to a specific policy. The frequency setting is performed by taking into account the current frequency setting, the thermal profile coming from on-die thermal sensors, and the workload requirement coming from the scheduler. For each functional unit $i = 1..p$, the workload is the minimum value of the clock frequency for executing the required tasks within the specified system constraints. The regulator provides a frequency assignment that minimizes the difference between the required and the achieved workload.

Several families of policies have been proposed in the literature for thermal management in MPSoC designs [Zanini et al. 2009b, 2009a, 2010a, 2010b]. All these policies have been developed and analyzed independently by developing specific problem formulations. In this work, we propose for the first time a comprehensive, unified theoretical formulation for the four main families of thermal management methods for MPSoCs. Thus, this article provides the analytical and experimental basis for selecting among the different state-of-the-art thermal control families based on the different requirements of the target MPSoC features (e.g., memory size, computing capabilities, etc.) and constraints (e.g., peak temperature, thermal balancing, etc.). Our exploration has shown that these different families are, in fact, closely related in the domain of *model predictive control* (MPC) [Bemporad et al. 2002], which has received much attention lately. In fact, all these policies aim at achieving an online smooth thermal control action that minimizes, to a certain degree, the performance loss according to different computational overhead and temperature observability accuracy constraints in the target MPSoC. Hence, the different control models included in these thermal management policies differ according to both the way architectural details (e.g., thermal models, scheduling, etc.) are included in the problem formulation and the way the solution is computed. On the one hand, distinctive key features in the problem formulation are the workload history information and the predictive nature of the algorithm. On the other

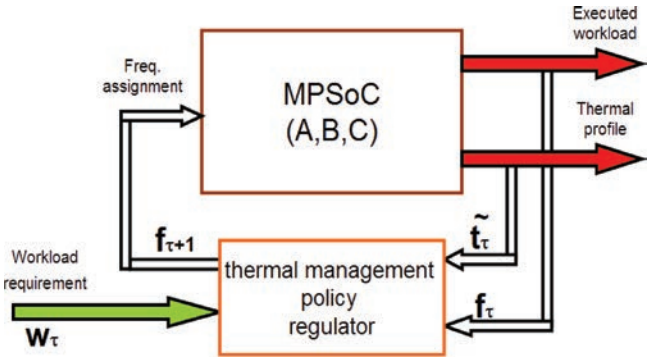


Fig. 1. Diagram of a generic DVFS-based thermal management system.

hand, distinctive key features in frequency assignment are the approximation error and the way the solution is computed: explicitly or implicitly (i.e., pre-computed and stored or calculated on the fly). To identify potential trade-offs among the compared state-of-the-art techniques, we implemented the analyzed policies on a unified MPSoC simulation platform. Results clearly show remarkably different trade-offs among the analyzed techniques and highlight the potentials of the methods discussed in this work for different types of thermal control requirements in MPSoC design.

The remainder of this article is organized as follows. Section 2 reviews the related state-of-the-art work on thermal control techniques for MPSoC. Section 3 describes the main thermal management policies under comparison. Section 4 describes the mathematical model to represent the MPSoC state and working conditions, whereas Section 5 presents our proposed unified thermal management formulation approach. Our target eight-core MPSoC design and the complete experimental setup are described in Section 6. Section 7 describes the results collected for the policies under comparison from the MPSoC simulations. Finally, Section 8 summarizes the main conclusions of this work.

2. RELATED WORK

Dynamic power management [Benini and De Micheli 1998] was developed first for single processors and later extended to MPSoCs [Donald and Martonosi 2006; Mukherjee and Memik 2006]. Most methods use *dynamic voltage and frequency scaling* (DVFS) for processor power optimization and balancing [Mukherjee and Memik 2006]. The following sections describe how DVFS has been embedded in different thermal management policies to manage both the performance and the thermal profile of the MPSoC.

Early methods based on monitoring the idle time in processors have been presented in Halfhill et al. [2000], where DVFS has been used to turn off functional units when they were not being used. Later, more refined control policies working at the OS level were proposed. Hanumaiah and Vrudhula [2012] and Xie and Hung [2006] present different sets of scheduling mechanisms for MPSoCs that perform temperature management at the system level. Donald and Martonosi [2006] obtain a significant reduction in localized hotspots using thread migration techniques. In Lu et al. [1999] present a software architecture that allows system designers to investigate power management algorithms in a systematic fashion. Coskun et al. [2007] present a dynamic scheduling algorithm with limited performance overhead. Simunic et al. [2004] discuss a methodology for managing power consumption in networks on chip. Similarly, Ogras et al. [2004] propose the control of power usage in processing elements (PEs) and routers by using models predictive control at design time, and Bogdan et al. [2012] elaborate further on

this approach by considering both PEs and routers in the control scheme for voltage and frequency. However, they only consider power management and do not explore thermal control aspects. Then, Merchant et al. [1996] present a variable speed processor that is thermally controlled based on an estimation on the hottest junction temperature of the chip. Bircher and John [2012] present processor counters that are used by the thermal policy to get an online monitoring of the overall power consumption.

These methods exploit neither thermal nor computational history information. The policy takes reactive decisions based on information related to the current thermal profile and frequency setting of the MPSoC to control. In several recent works, history information has been exploited to improve thermal management policies. Coskun et al. [2008a] exploits a temperature forecast technique based on an auto regressive moving average model. Lee et al. [2010] and Kang et al. [2011] propose policies for 2D and 3D MPSoC using autoregressive moving average applied to performance counters to find the correlation between the applied voltage setting and the measured MPSoC temperature. Coskun et al. [2008b] propose a novel technique that adapts the thermal management policy to the current workload characteristics. The adaptation is done online, exploiting information related to the workload history. Benini et al. [1999] and Qiu et al. [1999] present a stochastic power management system. These methods take power management decisions based on theories of discrete-time and continuous-time Markov decision processes and stochastic networks. Two recent approaches [Cochran and Reda 2010; Zhang and Srivastava 2010] describe two methodologies for achieving thermal prediction without completely relying on the thermal model—on thermal sensors and on power consumption statistical properties.

More advanced solutions proposed in the last years have applied different concepts of model-predictive control [Bemporad et al. 2002]. Wang et al. [2009] present a chip-level power control algorithm based on optimal control theory. This algorithm can control the power consumption of the MPSoC and can maintain the temperature of each core below a specified threshold. Magno et al. [2009] tailor a similar concept for multimodal video sensor nodes. In this article, we are considering and comparing four representative policies of these advanced solutions.

Previous policies do not completely avoid hotspots, but simply reduce their frequency. The main reason being that the interaction between the prediction method, the thermal behavior of the MPSoC, and the frequency assignment of the MPSoC has not been addressed as a joint optimization problem. The action taken by the policy to avoid hotspots does not address the problem from a global optimum perspective. This last point is the common problem of all previously mentioned policies, and it will be more clear in the following sections as the policies under comparison are described more in detail.

3. THEORETICAL ANALYSIS OF THERMAL MANAGEMENT POLICIES

In this section, we perform a comprehensive analysis of the four main families of thermal management policies for MPSoCs and compare their main features with respect to a wide set of metrics. In particular, the thermal policy techniques we consider are the linear quadratic regulator [Kang et al. 2011; Coskun et al. 2008a; Zanini et al. 2009b] (i.e., unconstrained MPC with horizon equal to infinity), the explicit/implicit model predictive control-based approach [Wang et al. 2009; Zanini et al. 2009a] (i.e., traditional MPC), the approximated explicit model predictive control policy [Zanini et al. 2010a] (i.e., approximated MPC), and finally, the convex optimization-based approaches [Hanumaiah and Vrudhula 2012; Zanini et al. 2010b] (i.e., joint workload and thermal profile prediction). This last technique is solved with a convex solver; however, it is an MPC as well with a linear objective function.

In order to develop a complete approach to exploring the complete design space of thermal management families, we have developed complete taxonomy and full

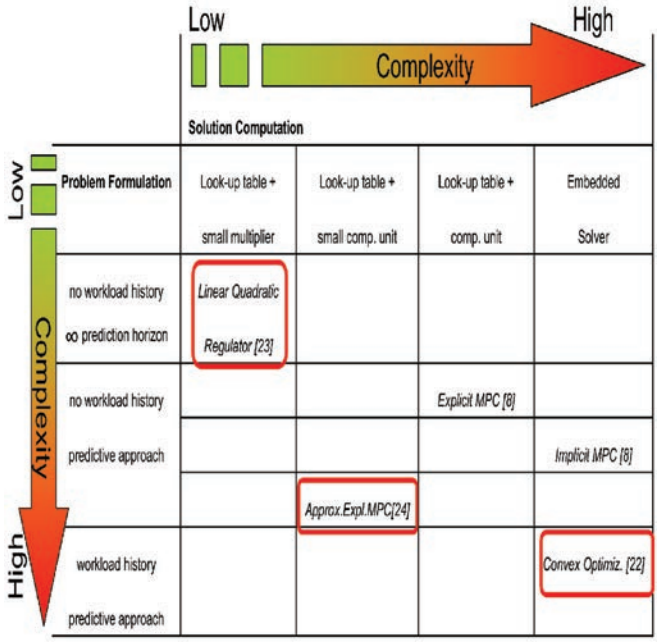


Fig. 2. Overview of compared thermal management policies.

comparison of the features of the main four state-of-the-art control families using MPC theory. Thus, we can perform a consistent classification regarding computation complexity, future forecasting precision, memory requirements, adaptability to different workload conditions at runtime, and smoothness in the peak temperature and thermal balancing control. In particular, the two fundamental metrics we used to differentiate the policies are complexity of the problem formulation and computational requirements to reach a stable control solution. The table on Figure 2 locates the compared policies graphically in this initial two-metric space.

The first row identifies thermal policies that do not take into account the workload history to forecast the future workload. For instance, although the entire formulation and solution of LQR is based on the idea of prediction, the prediction horizon (length of the prediction) on the temperature profile of the system is fixed and equals infinity, in this case. These policies only ensure that both power consumption and temperature spatial gradients are minimized. Their objective function targets the minimization of the difference between the target power consumption (required to get the work done) and the real one, as well as temperature spatial gradients minimization.

The *linear quadratic regulator* (LQR) does not allow one to set any threshold constraint on the MPSoC maximum temperature. However, a threshold-based mechanism is used to partially solve the problem.

The second row of Figure 2 identifies techniques that use prediction techniques to forecast the MPSoC thermal profile. The approximated, explicit, and implicit MPC belong to this family of policies. Their objective functions target the minimization of undone work. Then, the constraints considered in the problem formulation are the frequency range, the power-frequency relation, the thermal profile prediction horizon, and the maximum accepted temperature.

The last row identifies techniques performing a prediction on the MPSoC thermal profile by exploiting workload history information. An example is provided by convex

optimization. The objective function targets the minimization of three objectives: undone work, temperature gradient, and power consumption. Then, two additional constraints are considered with respect to the first and second row, namely, workload prediction and reliability of the workload prediction.

Each column of the table identifies a different level of computational requirement needed to compute online the solution to the problem formulation. The policies belonging to the first column require only a lookup table and a small multiplier to be implemented. The second column identifies techniques needing a lookup table and a small computational unit to be implemented. Here, the level of complexity is higher. For the third column, a lookup table and an average-to-large computational unit is required. Algorithms requiring an online solver belong to the last column.

The most relevant trade-off policies are highlighted by the red squares. They correspond to techniques taking into account the highest number of MPSoC properties for a given computational requirement. They indeed perform the best in making the problem formulation easy to solve, while exploiting the largest quantity of information in the determination of the MPSoC frequency assignment.

4. MPSOC MODEL

In this section, we describe the MPSoC model used in the problem formulation for thermal management and control. In particular, we propose models for the execution frequency versus power consumption relationship, the workload requirements constraints, and the transient heat flow propagation within the MPSoC physical layout.

4.1. Frequency Input Model

Using the MPSoC structure described in Figure 1, we model synchronous MPSoCs divided into p subsystems with p clocks that are viewed as the inputs to the system. Vector $\mathbf{f}_\tau \in \Re^{\mathbf{p}}$ represents the value of the clock frequencies at time τ. The frequency value of input i at time τ is $(\mathbf{f}_\tau)_\mathbf{i}$. Clock frequencies are continuous, ranging from $f_{\min}$ to a maximum frequency value $f_{\max}$, and represent our optimization variable. The previous statement is expressed by Equation (1).

$$\mathbf{f}_{\min} \preceq \mathbf{f}_\tau \preceq \mathbf{f}_{\max} \ \ \forall\, \tau, \tag{1}$$

where the symbol $\preceq$ means element-wise comparison, $f_{\max} \cdot \mathbf{1} = \mathbf{f}_{\max}$, and $\mathbf{1}$ is a vector of all ones of size p. The frequency vector represents our optimization variable. The value of its element is assigned by solving the minimization problem described in following sections. This minimization will try to achieve the desired performance requirements while completely satisfying given constraints.

At time τ, the relation between the normalized value of power dissipation $\mathbf{p}_\tau \in \Re^{\mathbf{p}}$ and the normalized frequency of operation $\mathbf{f}_\tau$ is expressed by Equation (2).

$$\mu \mathbf{f}_\tau^\alpha = \mathbf{p}_\tau \ \ \forall\, \tau, \tag{2}$$

where μ is a technology-dependent coefficient that adjusts the quadratic law of Equation (2) with real power values of the MPSoC in a way that minimizes the quadratic error between the two functions. The constant α depends on the technology as well, and usually takes a value between 1 and 2. If $\alpha = 1$, we have a linear dependence (i.e., frequency scaling), while if $1 < \alpha \leq 2$, we obtain a quadratic or subquadratic dependence (i.e., DVFS). In the case study we used in this work, we used $\alpha = 2$, because we assumed that most of the devices are working in velocity saturation [Rabaey 2009].

4.2. Workload Model

The workload is generated from higher-level software layers (e.g., operating system). In this article, we use a high-level abstraction of the workload. Using the MPSoC structure described in Figure 1, for each of p islands, the workload is defined as the

minimum value of the clock frequency that the functional unit should have to execute the required tasks within the specified system constraints.

The workload requirement at time τ is defined as a vector $\mathbf{w}_\tau \epsilon \Re^{\mathbf{p}}$, where $(\mathbf{w}_\tau)_\mathbf{i}$ is the workload requirement value for input i at time τ. In other word, the clock frequency that cores associated with input i from time τ to time $\tau+1$ should have in order to satisfy the desired performance requirement coming from the scheduler is denoted by $(\mathbf{w}_\tau)_\mathbf{i}$.

Our model is assumed to be continuous and ranging from zero to a maximum value $\mathbf{f}_{\max}$—the maximum frequency at which the cores can process data—namely, the following.

$$0 \preceq \mathbf{w}_\tau \preceq \mathbf{f}_{\max} \ \forall \tau. \tag{3}$$

When $(\mathbf{w}_\tau)_\mathbf{i} > (\mathbf{f}_\tau)_\mathbf{i}$, the workload cannot be processed, and so it needs to be stored and rescheduled in the following clock cycles. The way we measure the performance of the system in achieving the requested workload requirements at time τ is given by the vector $\mathbf{u}_\tau \epsilon \Re^{\mathbf{p}}$.

$$\mathbf{u}_\tau = \mathbf{w}_\tau - \mathbf{f}_\tau. \tag{4}$$

We call $\mathbf{u}_\tau$ the undone workload at time τ, and it expresses the difference at time τ between the requested workload and the workload that is actually executed by the MPSoC.

4.3. Heat Propagation Model

Our thermal model is based on finite-element analysis, as it has been commonly used by well-known system-level thermal analysis tools [Skadron et al. 2004; Atienza et al. 2008]. Two layers have been used in the vertical direction, namely, the silicon layer and the copper layer. The first one models the heat dissipation of integrated devices, while the second one models the metal wiring and the heat spreader. Though this model is simple, it is widely used in state-of-the-art works [Murali et al. 2008; Coskun et al. 2008a, 2008b]. Moreover, this modeling approach has been proven to be accurate enough for system-level thermal characterization in the case of 3D MPSoC with liquid cooling technologies [Coskun et al. 2010; Sabry et al. 2010].

The chip floorplan has been divided into several thermal cells of cubic shape, and each functional unit in the floorplan can be represented by one or more thermal cells of the silicon layer. Thermal modeling considers heat conductances and capacitances of the cells, as computed and validated in Skadron et al. [2004], Atienza et al. [2008], and Huang et al. [2008]. The discretization of the resulting differential equations can be solved using a discrete-time, linear, and time-invariant system [Zanini et al. 2009a].

The considered finite-element analysis for thermal modeling can be represented in a state-space form. In particular, in our case, by measuring all temperatures as offsets from the room temperature, the heat propagation process can be represented in the following way.

$$\mathbf{t}_{\tau+\mathbf{1}} = \mathbf{A}\mathbf{t}_\tau + \mathbf{B}\mathbf{p}_\tau, \tag{5}$$

$$\tilde{\mathbf{t}}_\tau = \mathbf{C}\mathbf{t}_\tau, \tag{6}$$

At time τ, the temperature of the next simulation step of cell i, that is, $(\mathbf{t}_{\tau+\mathbf{1}})_\mathbf{i}$ can be computed thanks to Equation (5). The relationship between the frequency assignment at time τ and the power consumption is expressed by Equation (2). Matrices $\mathbf{A} \epsilon \Re^{\mathbf{2n}\times\mathbf{2n}}$ and $\mathbf{B} \epsilon \Re^{\mathbf{2n}\times\mathbf{p}}$ describe the heat propagation properties of the MPSoC. The total number of cells in the silicon plus the copper layer is $2n$. Equation (6) describes the choice on the number of temperature sensors inside the MPSoC, which needs to be at least equal to the minimum number of terms of the state-space form for each MPSoC to have a full visibility of the temperature. Then, matrix $\mathbf{C} \epsilon \Re^{\mathbf{s}\times\mathbf{2n}}$ relates the temperature value

of each cell to the temperature measurement of a particular sensor. In this model, we realistically assume that only a limited number of thermal sensors exists in the final MPSoC, thus, the temperature can be measured only in that limited number of locations or thermal cells. This number is different for each considered MPSoC, but the proposed formulation is still valid and applicable.

In the model described in the previous paragraph, a state is required for every block composing the floorplan. The reason being that we need $2n$ states to store n temperatures for each of the 2 layers—each one composed of n cells. This requirement is expensive in terms of computational requirements for high-accuracy MPSoC models. The state of the system, to avoid inaccuracy mismatches between the model and the real MPSoC, is estimated from real-time measurements. The larger the number of states modeling the MPSoC, the larger the number of sensors required for its state estimation.

Indeed, as the model requires a state for every block composing the floorplan, it is expensive in terms of computational requirements for high-accuracy thermal modeling of MPSoC designs. Hence, to reduce the number of states, we perform a model order reduction using a Gramian-based balancing of state-space realizations [Laub et al. 1987]. We have selected this technique because it can perform an equivalent transformation of the states of the system in order to produce a new set of modes ordered by their importance. Therefore, in the context of thermal modeling, the modes with negligible contribution to the input-output response (i.e., thermal response) are deleted by enforcing the matching of the steady-state thermal response (DC gains of the RC thermal network representation) between the original system and the reduced one. Thus, the number of states of the new reduced order model depends on the specific MPSoC being modeled, and in our formulation, this number (l) is parameterized without any loss of generality. Therefore, we can now describe the full MPSoC thermal model in a general way using the following system of equations.

$$\mathbf{x}_{\tau+\mathbf{1}} = \tilde{\mathbf{A}}\mathbf{x}_{\tau} + \tilde{\mathbf{B}}\mathbf{p}_{\tau}; \tag{7}$$

$$\tilde{\mathbf{t}_{\tau}} = \tilde{\mathbf{C}}\mathbf{x}_{\tau}, \tag{8}$$

with matrix $\tilde{\mathbf{A}} \in \Re^{l \times l}$ and matrix $\tilde{\mathbf{B}} \in \Re^{l \times p}$. The number of states of the new thermal model is l, and p is the number of frequency island in the MPSoC. Then, Equation (7), which is analogous to Equation (5), describes the state update for the reduced order model of the MPSoC. However, in this case, the states do not directly represent temperature values inside a cell, but accumulated thermal figures in groups of them.

Matrix $\tilde{\mathbf{C}} \in \Re^{s \times l}$ in Equation (8) relates the value of the states to temperature measurements in s specific locations inside the MPSoC. This equation is analogous to Equation (6) and describes how the temperature measurements can be derived from the state vector $\mathbf{x}$. To this end, the thermal model is adjusted dynamically based on the precise on-chip temperature measurements of the actual MPSoC thermal sensors. Thus, on-chip thermal sensors are used to recalibrate at runtime for possible imprecisions of the MPSoC thermal model, which produces a fully accurate MPSoC thermal management scheme.

5. UNIFIED PROBLEM FORMULATION FOR THERMAL MANAGEMENT FAMILIES

All the main classes of policies for thermal control target the fulfillment of a common set of objectives in order to give bounds on MPSoC runtime performance and to prevent thermal runaway due to thermal control instabilities. First, they try to ensure that the maximum MPSoC temperature never exceeds a predefined threshold. Second, they try to avoid abrupt frequency and temperature variations, both over time and over space. Third, their objective is also to minimize the requested (but not executed) work from the scheduler. Thus, the control problem to fulfill all these requirements can be formalized

in the following way.

$$J = \sum_{\tau=1}^{h} \left(\|\mathbf{Q}\mathbf{x}_\tau\|_{\mathbf{g}} + \|\mathbf{R}\mathbf{p}_\tau\|_{\mathbf{j}} + \|\mathbf{T}\mathbf{u}_\tau\|_{\mathbf{b}} \right). \tag{9}$$

$$min \ J; \tag{10}$$

$$subject\ to: \quad 0 \preceq \mathbf{f}_\tau \preceq \mathbf{f}_{\max} \ \forall \tau; \tag{11}$$

$$\mathbf{x}_{\tau+\mathbf{1}} = \tilde{\mathbf{A}}\mathbf{x}_\tau + \tilde{\mathbf{B}}\mathbf{p}_\tau \ \forall \tau; \tag{12}$$

$$\tilde{\mathbf{C}}\mathbf{x}_{\tau+\mathbf{1}} \preceq \mathbf{t}_{\max} \ \forall \tau; \tag{13}$$

$$\mathbf{u}_\tau \succeq \mathbf{0} \ \forall \tau; \tag{14}$$

$$\mathbf{u}_\tau = \mathbf{w}_\tau - \mathbf{f}_\tau \ \forall \tau; \tag{15}$$

$$\mathbf{p}_\tau \succeq \mu \mathbf{f}_\tau^\alpha \ \forall \tau. \tag{16}$$

Function J is expressed by three sums, where the summation index τ ranges from 1 to h. During these h future steps the system tries to minimize the cost function J (Equation (10)) and computes the frequency assignment for these steps.

The first term $\|\mathbf{Q}\mathbf{x}_\tau\|_{\mathbf{g}}$ is the g norm of the state vector x weighted by matrix $\mathbf{Q}$. Matrix $\mathbf{Q}$ is related to hotspot minimization and thermal balancing, and its coefficients can be tuned to give the relative important to one, with respect to the other factor. The importance will be tuned accordingly for the different considered thermal control policies, as shown in Section 6.4; otherwise, in our experiments we consider an equal importance to both factors. The second term $\|\mathbf{R}\mathbf{p}_\tau\|_{\mathbf{j}}$ is the j norm of the input power vector p weighted by matrix $\mathbf{R}$. Matrix $\mathbf{R}$ is diagonal and quantifies the importance that power minimization has in the optimization process. The third term $\|\mathbf{T}\mathbf{u}_\tau\|_{\mathbf{b}}$ is the b norm of the amount of predicted required workload that has not been executed. Matrix $\mathbf{T}$ is diagonal and quantifies the importance that executing the workload required from the scheduler has in the optimization process.

Then, as optimization constraints in the interval with the same range as the summation index τ, Equation (11) first defines the range of working frequencies that can be used. It enables a continuous range of frequency settings, but this does not prevent the optimization problem from adding in a limitation on the number of allowed frequency values. Then, Equation (12) defines the evolution of the system according to the present state and inputs. Next, Equation (13) states that temperature constraints should be respected at all times and in all specified locations. The maximum allowed temperature is $T max$. Then $\mathbf{t}_{\max} = T max \cdot \mathbf{1}$. Since the system cannot execute jobs that have not arrived, every entry of $\mathbf{u}_\tau$ has to be larger than or equal to 0, as stated by Equation (14). The undone work at time τ, u_τ, is subsequently defined by Equation (15). To this end, the relation between the power vector $\mathbf{p}$ and the working frequencies is expressed by Equation (2). Nonetheless, in Equation (16), we relax Equation (2) to an inequality. Overall, it can be proven that the resulting relaxed convex problem is equivalent to the original problem with the quadratic equality constraint (cf. [Boyd and Vandenberge 2004], p. 191).

The control problem is formulated over an interval of h time steps, which starts at current time τ. For this reason, the approach is said to be predictive. The result of the optimization is an optimal sequence of future control moves (i.e., frequency settings for the cores). Only the first sample of such a sequence is actually applied to the process; the remaining moves are discarded. At the next time step, a new optimal control problem based on new temperature measurements and required frequencies is solved over a shifted prediction horizon. Such a receding-horizon [Bemporad et al. 2002] mechanism represents a way of transforming an open-loop design methodology into a feedback

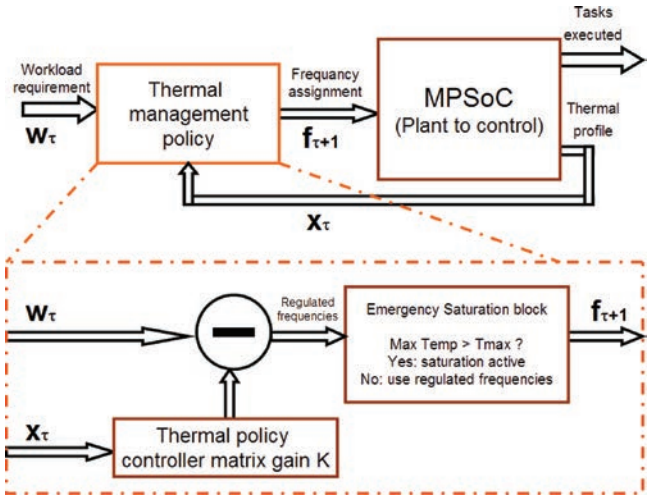


Fig. 3. Linear quadratic regulator-based policy block diagram.

one, as at every time step, the input applied to the process depends on the most recent measurements.

It is important to notice that the proposed problem formulation is applied to various types of MPSoC architectures. User-defined parameters contained in matrices $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{T}$, the prediction horizon h, and the norms indices g, j, and b have influence on the performance of the implemented policy. These parameters indeed drive the optimization target trade-offs between power minimization, satisfaction of user requirements, and thermal gradients.

5.1. Linear Quadratic Regulator

The *linear quadratic regulator* targets temperature and power difference minimization using a linear discrete time system.

We next show how the generic model can be specialized to be a LQR. When the maximum MPSoC temperature is less than a certain threshold $Tmax$, the overall system (presented in Figure 3) is a linear feedback system, where MPSoC frequencies are calculated simply by subtracting from the workload requirement $\mathbf{w}_\tau$, the product of the state vector $\mathbf{x}_\tau$ and the controller matrix gain $\mathbf{K}$. The state vector $\mathbf{x}$ is related to the thermal profile by Equation (8). The emergency saturation block (in Figure 3) just saturates the regulated frequencies to a certain value when the maximum MPSoC temperature is higher than the threshold $Tmax$. This allows the MPSoC to cool down and to reduce its maximum temperature in case of overheating.

5.1.1. Problem Formulation. By looking at the general model of Equations (9)–(16), the problem can be formulated in the following way.

The horizon is infinite, and the reference (the requested workload $\mathbf{w}_\tau$) is assumed to be constant over all this period. Matrix $\mathbf{T}$ is a zero matrix, and the norm $g = j = 2$. Thus, the objective function in this case is a squared 2-norm (i.e., a quadratic function). Since this method is defined only for linear regulation, it is not possible to include any constraint inside the problem formulation. For this reason, no frequency, temperature or undone work constraint can be added. However, all these constraints are considered after the optimization. Frequencies outside the frequency range are discarded. A bias signal $\mathbf{w_t}$ is added to the control loop to force the system to execute the requested

workload $\mathbf{w_t}$. Finally, an emergency saturation block is added to avoid the maximum MPSoC temperature being higher than the threshold $T max$.

5.1.2. Design Phase. The state feedback controller gain $\mathbf{K}$ presented in Franklin et al. [1997] can be computed using the infinite horizon solution of the discrete-time Riccati equation associated with this system. The solution exists only if matrix $\mathbf{Q}$ is positive semi-definite and matrix $\mathbf{R}$ is positive definite (i.e., identity matrix). According to the way matrix Q and R have been defined (see Section 5), they always fulfill the requirements. While the original system is passive, the reduced system is not necessarily passive. However, the reduced system is stabilizable, since it is passive and since balanced reduction retains stabilizability. Consequently, all LQR controllers get properly stabilized in the case of MPSoC floorplans that we are considering [Franklin et al. 1997].

5.1.3. Runtime Phase. If the MPSoC maximum temperature is under a predefined threshold during the online optimization phase, the optimum frequency assignment to achieve thermal balancing is calculated using the following equation.

$$\mathbf{f}_{\tau+\mathbf{1}} = \mathbf{w}_\tau - \mathbf{K} \cdot \mathbf{x}_\tau, \tag{17}$$

where at time τ, $\mathbf{x}_\tau$ is the current state and $\mathbf{w}_\tau$ is the workload required to fulfill performance requirements. The number of multiplication and additions N_{op} required every time the policy is applied at runtime is given by the following equation.

$$N_{op} = l \cdot p, \tag{18}$$

where l is the dimension of the state vector and p is the number of cores of the system.

5.2. Explicit/Implicit MPC

In general, MPC is an optimal control approach aimed at maximizing a performance metric for a linear dynamic system under input/output constraints. The solution of the optimization problem provides the feedback control actions that will determine the frequency setting of the following clock cycles. The policy can be implemented by embedding a numerical solver in the real-time control code (implicit solver), or can be precomputed offline and evaluated through a lookup table of linear feedback gains (explicit solver). Figure 4 shows the block diagram of an explicit MPC-based policy. The *plant to control* is the MPSoC that we want to control. In the case of implicit MPC, the orange dotted box is implemented using an embedded solver that solves online the frequency assignment problem. The block diagram is shown in Figure 5.

5.2.1. Problem Formulation. According to the general model of Equations (9)–(16), the formulation is the following.

The horizon is finite and equal to h, and the reference (the requested workload $\mathbf{w}_\tau$) is assumed to be constant over all this period. The state and the power cost function matrices $\mathbf{Q}$ and $\mathbf{R}$ are set to be null matrices. Matrix $\mathbf{T}$ is the identity matrix and the norm $b = 2$. Thus, the objective function is a quadratic form that minimizes the undone workload. All the others constraints expressed by Equations (11)–(16) are considered inside the problem formulation.

5.2.2. Implicit versus Explicit Regulator. The proposed control strategy can be implemented in two different ways. The first one is called implicit and requires solving the minimization problem online every time the policy is applied. Thus, a significant amount of hardware resources are needed, since the result must be computed in a time frame shorter than the thermal time constants of the MPSoC.

An alternative approach has been proposed [Bemporad et al. 2002]. In this case, the problem is solved offline in a way that makes explicit the dependence of the solution to the frequency assignment problem $\mathbf{f}_\tau^\alpha$ on input vectors $\mathbf{f}_\tau^\alpha$, $\mathbf{w}_\tau^\alpha$, and $\mathbf{x}_\tau$. The state space

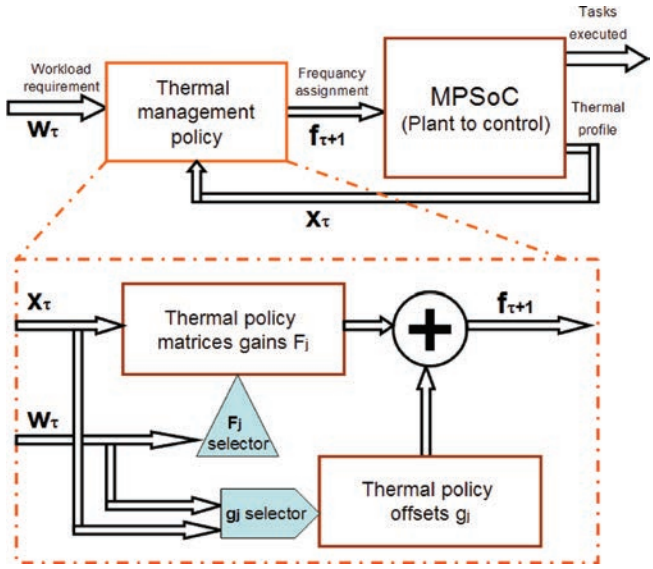


Fig. 4. Explicit model predictive control-based policy block diagram.

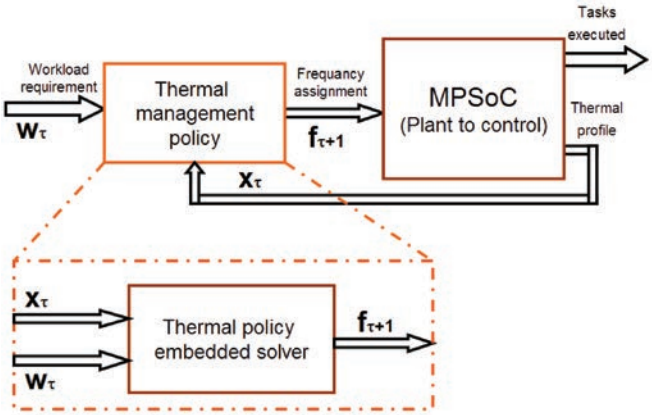


Fig. 5. Embedded solver-based policy block diagram.

can be divided into a set of regions bounded by linear inequalities (i.e., a polytope), and in each region, a different linear controller can be specified and computed offline.

Then, the controller selection can be efficiently performed online by simply checking region boundaries. The resulting controller structure is defined in any of the M partitions as follows.

$$\left[\mathbf{f}^{\alpha}_{\tau+1}\right] = \mathbf{F_j}\begin{bmatrix}\mathbf{x}_\tau\\ \mathbf{f}^{\alpha}_{\tau}\\ \mathbf{w}^{\alpha}_{\tau}\end{bmatrix} + \mathbf{g_j} \;\; \textbf{if} \;\; \mathbf{H_j}\begin{bmatrix}\mathbf{x}_\tau\\ \mathbf{f}^{\alpha}_{\tau}\\ \mathbf{w}^{\alpha}_{\tau}\end{bmatrix} \leq \mathbf{k_j}, \tag{19}$$

where matrix $\mathbf{F_j}$ and vector $\mathbf{g_j}$ are the gain and offset coefficients of the j^{th} region (cf. Figure 4). Each region is identified by affine inequalities defined by the matrix $\mathbf{H_j}$ and vector $\mathbf{k_j}$ in Equation (19) (see [Bemporad et al. 2002] for more details).

If the partitions are properly stored, the number of operations depends logarithmically on the partitions [Tondel et al. 2003]. Nonetheless, while the computer code for evaluating MPC in the explicit form is certainly simpler than the code embedding the QP solver, from the point of view of memory requirements, the explicit form may be more demanding, as M and the matrices to be stored in lookup tables may be large.

The state dimension and the length of the prediction horizon h affects the number of regions. There is indeed (in a worst-case scenario) an exponential relation between these last two parameters. However, the longer the prediction horizon is, the better the policy performance is in most cases. Thus, for large horizons, an implicit implementation may be more convenient in terms of power dissipation and area consumption than an explicit one or viceversa.

According to Equation (19), the number of coefficients N_c to implement the optimum explicit approach of our method is given by the following expression.

$$N_c = N_{reg} \cdot (\overbrace{2p(p+l/2)}^{\mathbf{F_j}} + \overbrace{p}^{\mathbf{g_j}} + \overbrace{2N_{avg} \cdot (p+l/2)}^{\mathbf{H_j}} + \overbrace{N_{avg}}^{\mathbf{k_j}}). \tag{20}$$

N_{reg} is the number of regions of the explicit controller, p is the number of processing cores, and l is the dimension of the state vector $\mathbf{x}$. N_{AVG} is the average number of affine inequalities that identifies each region. In this equation, the symbol $\overbrace{\ }$ highlights contributions for each of the matrices in Equation (19).

The computational effort of the evaluation can be computed from Tondel et al. [2003] and Equation (19) and, assuming an average-case scenario, is provided by the following equation.

$$N_{comp} = 1.7 \cdot log_2(N_{reg}); \tag{21}$$

$$N_{Mult-Add} = 2p(p+l/2), \tag{22}$$

where N_{comp} is the number of required comparisons and $N_{Mult-Add}$ is the number of required multiplications/additions.

5.3. Approximated Explicit Predictive Policy

This method solves with an approximate algorithm the region partitioning of the optimum explicit approach presented in Section 5.2. This method has similarities to the explicit approach, and both its block diagram and problem formulation are shown in Figure 4. Compared with the optimum approach (see Section 5.2), it provides a significant reduction in hardware requirements and computational cost at the expense of a small loss in accuracy. This enables an increase of the prediction length and the accuracy of the MPSoC model and, therefore, the performance of the method. This policy is based on the work of Zanini et al. [2010b].

The solution to the optimization problem is computed offline in a way that makes explicit the dependence of the solution to the frequency assignment problem $\mathbf{f_{\tau+1}}$ on input parameters $\mathbf{w}_\tau$ and $\mathbf{x}_\tau$. The resulting explicit controller is *piece-wise polynomial*. In other words, the state space can be divided by a set of regions bounded by linear inequalities (i.e., a polytope), and in each region, a different polynomial controller can be specified and computed offline [Jones and Morari 2010]. Then, the controller selection can be efficiently performed online by simply checking region boundaries.

5.3.1. Approximation Algorithm. The proposed method begins by computing an approximate convex *piece-wise affine* (PWA) lower bound of the optimal cost function J described in Equations (9)–(16). Since the approach proceeds in an incremental greedy-optimal fashion, it is possible to stop the process when any desired level of complexity, or approximation accuracy, is reached. The control law is then derived from this lower bound using the barycentric technique proposed in Warren et al. [2007]. The result is

a nonlinear and smooth piece-wise polynomial control law. In particular, the algorithm is divided into two main phases.

In this algorithm, we first compute the approximation level by selecting a number of regions to distinguish in the thermal response and control of the system. Then, the bounds of that particular region are extended with the thermal response corner cases so that the approximation level is maximally reduced in the considered region and the whole design space is covered. It can be shown that a specific desired approximation error can be achieved in finite time for any convex function [Boyd and Vandenberge 2004]. The final result is a smooth and piece-wise polynomial thermal control approach.

5.4. Convex Optimization-Based Policies

Figure 5 shows the block diagram of this family of policies. In this case, there is an embedded solver that computes online the problem formulation.

5.4.1. Problem Formulation. According to the general model of Equations (9)–(16), the formulation is the following.

The horizon is finite and equal to h. The state cost matrix $\mathbf{Q}$ is set to be a null matrix. Matrix $\mathbf{R}$, responsible for the power minimization, is an identity matrix, and the norm $j = 1$. All the other constraints expressed by Equations (11)–(16) are considered inside the problem formulation.

The major differences with respect to previous problem formulations is due to the fact that in this case, the predicted profile is asummed to be time-varying, whereas before it was considered constant. Therefore, in this case, matrix $\mathbf{T}$ is time varying, the norm $b = 1$, and the reference (i.e., the requested workload $\mathbf{w}_\tau$) is time varying as well. The algorithm indeed dynamically predicts the future workload requirements $\mathbf{w}_\tau$, and the reliability of the estimation is embedded in the problem formulation by matrix $\mathbf{T}$. Matrix $\mathbf{T}$ relates to the accuracy of the estimation and it is updated online according to the predicted workoad. The more the prediction error, the less the undone work will be considered in the optimization process. Thus, the better the prediction, the higher the weight on the undone workload for time horizons $\tau > 1$ far from the current one of $\tau = 0$.

5.4.2. Workload Prediction and Its Reliability. To increase the performance of the proposed policy, history information about the task arrival process is exploited by the algorithm. These data are used to make predictions on future workload requirements. Reliability information on how good the prediction is are also considered in the optimization process.

The prediction is done by using a linear model to perform a best d^{th} order polynomial fit. This polynomial fit is performed by minimizing the error within the observed window of temperatures by using the following function.

$$\|\mathbf{w}_\tau - \check{\mathbf{A}}\mathbf{e}_\tau\|_{\mathbf{2}}^{\mathbf{2}}, \tag{23}$$

where $\mathbf{w}_\tau$ contains the frequency requirements $\forall \tau = 1..N$, where N is the length of the observation window of historical data. Vector $\mathbf{e}_\tau \in \Re^{\mathbf{d+1}}$ and $\check{\mathbf{A}} \in \Re^{d+1,N+h}$ are used in the polynomial interpolation process. Equation (23) can be solved as a least squares minimization problem to derive vector $\mathbf{e}_\tau$. The prediction on the future workload requirement is performed by assuming that the linear model just derived will hold for the next h data samples. With this assumption, the future workload requirement is given by following equation.

$$\hat{\mathbf{w}}_\tau = \check{\mathbf{A}}\mathbf{e}_\tau, \quad \forall \ \ \mathbf{N} \leq \tau \leq \mathbf{N} + \mathbf{h}, \tag{24}$$

where $\hat{\mathbf{w}}_\tau \in \Re^p$ for $\tau > N$ is the predicted workload requirement at time τ.

The way we take into account the accuracy of this prediction is embedded in the weighting matrix $\mathbf{T}$. This matrix is chosen according to the reliability of the workload

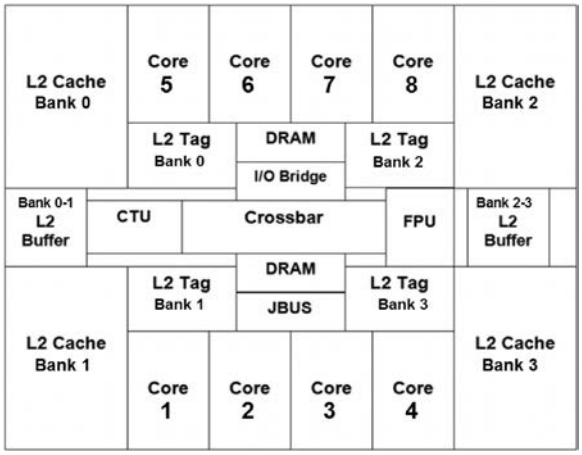


Fig. 6. Floorplan used of the Niagara-1 multi-core case study.

prediction. We have chosen these parameters to achieve a good prediction according to empirical studies performed on different benchmarks [Coskun et al. 2008a] for representative examples of the MPSoCs under study in this work.

6. EXPERIMENTAL SETUP

In this section, we summarize the target MPSoC architecture, where the different families of thermal control algorithms are evaluated, as well as how we interface the cycle-accurate MPSoC simulator of the target architecture with the various control modules we have developed to implement the different algorithms.

6.1. MPSoC Model and Workloads

The MPSoC architecture we are considering models the eight-core Niagara-1 (UltraSparc T1) architecture from Oracle [Kongetira et al. 2005]. The floorplan of the Niagara-1 multi-core architecture is presented in Figure 6. The size of the state vector after the model order reduction is 12 elements. Thus, the minimum amount of sensors to fully characterize the Niagara-1 floorplan is 12. However, in order to evaluate the degradation of the thermal measurements due model order reduction, we have evaluated configurations ranging from this minimum number of sensors up to having one sensor per element in the silicon layer, namely, a total of 30 sensors. Our results indicate that there is a negligible benefit in accuracy when more sensors are used. Similarly, the floorplan has been modeled using blocks of $3mm$ side each, and in case a specific floorplan unit has a larger size, it has been decomposed into multiple blocks. Then, the values of technological parameters and coefficients have been derived from Sabry [2005] and Kongetira et al. [2005]. Silicon conductivity and capacitance are, respectively, $130W/(m \cdot K)$ and $1635660J/(m^3 \cdot K)$. Wiring layer conductivity and capacitance are, respectively, $2.25W/(m \cdot K)$ and $2174502J/(m^3 \cdot K)$. The heat spreader resistance is set to $2W/^{\circ}K$. This architecture has a maximum operating frequency of 1.2GHz, and the maximum power consumption of each processor core at this frequency is 4W [Kongetira et al. 2005]. The interconnect power consumption for each block has been assumed to be proportional to the frequency setting of that specific block. To implement the voltage and frequency scaling techniques, we use frequencies ranging from 171MHz to 1.2GHz. In this range, only specific values of frequencies are allowed;

Table I. Comparative Table of Workload Characteristics

	Benchmark	Average CPU Load	L2 I-Miss	L2 D-Miss	FP unit
1	Web-med	53.12%	12.9	167.7	31.2
2	Web high	92.87%	67.6	288.7	31.2
3	Database	17.75%	6.5	102.3	5.9
4	Web+DB	75.12%	21.5	115.3	24.1
5	gcc	15.25%	31.7	96.2	18.1
6	gzip	9	2	57	0.2
7	Mplayer	6.5%	9.6	136	1
8	Mplayer+Web	26.62%	9.1	66.8	29.9

thus, the values for the frequencies are expressed by Equation (25).

$$f = \frac{6 \cdot 10^9}{D_f} \quad \forall\, D_f,\ subj.\ to: \quad 5 \leq D_f \leq 35, \tag{25}$$

where D_f is the integer division factor needed by the clock tree generator.

During the execution of the policies at runtime, the frequency from continuous to discrete-time is mapped according to a *round half up strategy*. Thus, the processor can operate at a higher frequency than the required one in order to meet the current workload deadline. In this case, the processor would be dissipating more energy than the theoretical minimum required to execute a specific workload.

We modeled the different workloads of a mix of different benchmarks, ranging from Web-accessing to playing multimedia [Coskun et al. 2008a, 2008b]. To this end, we analyzed the percentage of CPU load, I/O utilization, and memory access rates of these different workloads, and we implemented in the job scheduler the different observed workloads features by varying the load of a set of large multiplication matrices and memory accesses intervals. The recorded execution characteristics are shown in Table I. Moreover, using cpustat, we recorded the cache misses and floating point (FP) instructions per 100K instructions in our cycle-accurate MPSoC simulation framework (Section 6.1), and they are also reported in Table I.

The average CPU load is 62% and ranges during the runtime execution from 10% to 100%. Jobs have an average duration of $2ms$ but oscillate from $0.1ms$ to $10ms$. The experiments are conducted using a large trace with around 60,000 tasks, which include the behaviors of the workloads shown in Table I, in order to have several hundreds of seconds for realistic thermal behavior simulation during actual system execution. As an initial condition, we assumed the system to be in off state and at room temperature ($300^\circ K$).

We compute the leakage power of processing cores and functional units as a function of their area and specific temperature. We assume a base leakage power density of $0.25W/mm^2$ at $383^\circ K$ for $90nm$, as in Bose [2003]. T_o accounts for the temperature effects on leakage power, and we use the model proposed in Coskun et al. [2010]. In this case, the leakage power at a temperature T_o, $^\circ K$, is given by: $P(T) = Po \cdot e^{\beta(T-383)}$, where P_o is the leakage power at $383^\circ K$ and β is a technology dependent coefficient. Finally, we set $\beta = 0.017$ [Sabry et al. 2010]. For the techniques under comparison, the error by neglecting the leakage power in the problem formulation is less than 0.3 degrees (see Figure 10).

The cycle-accurate thermal simulation framework we extended to validate the results presented in this work is described in Atienza et al. [2008] and Murali et al. [2008]. This framework includes a validated thermal model for 90nm technology using a 3D finite element method simulator. We have chosen this framework due to the high thermal accuracy achieved by this framework due to its previous calibration for the target

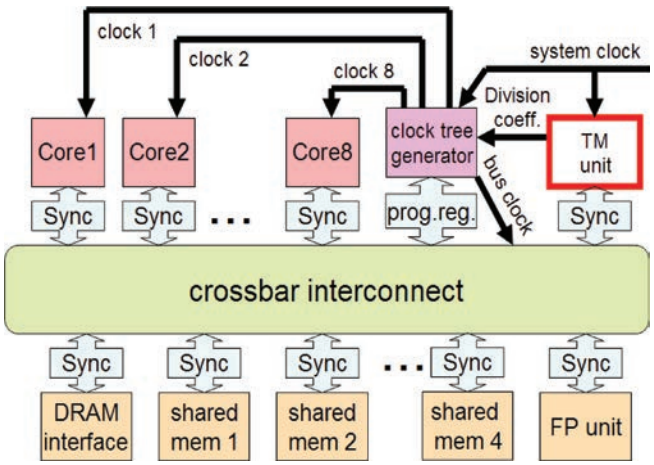


Fig. 7. Simulation infrastructure block diagram.

MPSoC architecture technology versus other (more generic) state-of-the-art MPSoC thermal simulation approaches.

6.2. Virtual Platform Environment and DVFS Support

The block diagram of our Niagara-1 MPSoC (UltraSparc T1 [Kongetira et al. 2005]) simulation infrastructure is shown in Figure 7. The simulation architecture is a SystemC-based simulation platform. The main device consists of eight 32-bit cores, and four shared memories (L2 Cache Banks(0..3)). Private memories are assumed to be inside each core (by core, we refer to a processing unit and a private memory). All these units communicate with each other by a crossbar interconnect. A floating point unit is also connected to it. A DRAM interface connects to main memory. The virtual platform environment also provides power statistics for the several hardware modules in the simulated platform. The simulation is based on real applications generating functional data traffic on the target architecture.

The virtual platform supports different working frequencies and voltages for each processor core. For this purpose, a variable clock tree generator (see Equation (25)), programmable registers, and a synchronization module have been integrated in the simulation platform. The clock tree generator feeds the hardware modules of the platform (processors, memories, etc.) with independent and frequency-scaled clock trees. The frequency-scaled clock trees are generated by means of frequency dividers (shift counters), whose delay can be configured by users at design time. The scheduling algorithm simply gives the queuing job to the first available functional unit that has enough resources to execute it.

Scaling the clock frequency of the processors creates a synchronization issue with the system bus, which is assumed to work at the maximum frequency. The processing cores and the bus interface communicate by means of a handshaking protocol which assumes the same working frequency at both sides. Therefore, a synchronization module was designed, containing two dual-ported FIFOs wherein data and addresses sent by the bus interface to the processor and vice versa are stored. This module works with a dual clock: one feeding the core side and one feeding the bus interface side. Finally, as shown in Figure 7, the module also takes care of properly interfacing processor to bus signals, and a dedicated sub-module is implemented for this purpose.

6.3. Support for Thermal Management Policies

To simulate thermal management policies, we added a *thermal management* (TM) unit to the network of Figure 7. This unit is a SystemC module that supports intercommunication between the thermal policies written in Matlab with the SystemC-based platform. This collects information related to current runtime power consumption, freezes the SystemC simulation, and starts a Matlab-based function that implements the thermal management policies. Just collected information is passed to this function. The Matlab function analyzes these data and solves the frequency assignment problem. The solution represents the frequency setting for the next cycles, until the thermal policy is applied again after T_{pol} seconds. Once the solution is computed, the TM unit unfreezes the simulation and waits a specific amount of time T_{cp} before setting frequencies and voltages among the cores and the interconnect. T_{cp} models the time needed by the actual hardware implementation of the policy to solve the frequency assignment problem.

In our case, we assumed a worse case scenario by setting T_{cp} equal to $50\mu s$. In this case, to make this assumption real, we assume that all policies are executed in optimized software implementations and that all the frequency transition overheads are included. Optimized implementations of this software take a few microseconds (for the case study described in the experimental set up section) to run on a state-of-the-art computing machine [Eldar et al. 2010; Boyd and Wegbreit 2007], which is at least three orders of magnitude less than the interval between two consequent applications of the policy (from 10ms to 100ms). For the previously mentioned reasons, most of the T_{cp} re-synchronization penalty is related to the DVFS setup time. However, according to the technology used, it is possible to increase or decrease this time to any value from 0 to the time between two consecutive applications of the policy. In this work, we were mostly focused on comparing the policies rather than optimizing them with an extremely fine granularity tuning of all their parameters.

6.4. Policy Setups

In all our experiments, the thermal management policies are applied every $T_{pol} = 10ms$, while the simulation step for the discrete time integration of the RC thermal model (Section 3) has been set to 200μs. The maximum temperature limit is set to $370^{\circ}K$. The room temperature is set to $300^{\circ}K$. In the problem formulation, we used $\alpha = 2$ [Murali et al. 2008] to establish the relation between the frequency setting and the power consumption, because we assumed that devices are working in velocity saturation [Rabaey 2009].

—*No thermal control (no policy).* In this case, there is no thermal control strategy being applied to the MPSoC under consideration. Thus, the MPSoC operates always at the required frequency to optimize performance according to the required workload at each moment in time, without considering any possible temperature constraint. Although this is not a realistic situation, we include it in our experiments as the ideal situation in case no thermal concerns were to outline the severity of the thermal problem.

—*Threshold-based DVFS (TB-DVFS).* This policy sets the frequencies of the cores according to the requirements coming from the scheduler. If the maximum chip temperature goes above a specified threshold (in this case study, set to $370^{\circ}K$), the policy sets the frequency up to 62.5% of the maximum [Murali et al. 2008].

—*Linear quadratic regulator (LQR).* This setting tries to keep a uniform thermal profile rather than minimizing power consumption. Thus, in the problem formulation, we minimize thermal unbalance while respecting the power limit. According to our experimental model, where the number of cores p equals 8 and n equals 30, the

Table II. Table of MPC-Based Thermal Approaches Having Different Approximation Complexities

Controller	Number of Regions	Number of Vertices	Comp. effort (#operations)			Storage Space (#coefficients)			Time Design [s]	Time Run [ms]
			Search	Control Law	Total	Search	Control Law	Total		
MPC-100	1	66	0	858	858	0	366	366	40.13	4.29
MPC-200	14	136	30	1,025	1,055	204	915	119	46.92	5.27
MPC-300	32	233	50	1,184	1,234	1560	1,770	3,330	50.75	6.17
MPC-400	51	328	60	1,286	1,346	3186	2,778	5,964	55.90	6.73
MPC-500	72	431	60	1,335	1,395	4986	3,771	8,757	63.61	6.97
MPC-600	87	528	65	1,354	1,419	7212	4,683	11,859	71.46	7.09
MPC-opt	3,770		89,552	16	89,568	89,552	60,320	149,872	196.32	447.84

number of required coefficients to store multiplications and additions equal to $30 \cdot 8 = 240$. These operations need to be performed every T_{pol}.

—*Model predictive control-based policy (MPC).* To design the regulator, we used a Matlab-based development platform provided by Kvasnica et al. [2004]. Table II compares the approximated approach with the optimum approach proposed in Zanini et al. [2009a]. With the optimum approach, we refer to the control policy presented in Section 5.2, while with the approximated approach, we refer to the control policy presented in Section 5.3. The resulting polynomial control laws range, from 100 to 600 vertices. The penultimate column reports the time in seconds that a MacBook Pro (2.8GHz, Core 2 Duo, 4GB RAM) took to design the controllers. By comparing with the optimal controller, we get a reduction in the computation time ranging from 2.7x to 4.9x. The last column reports the time needed to runtime execute the policy, assuming a processor able to execute $200K$ FLOPs. A reduction versus the optimum approach ranging from 63.1x to 104.4x can be achieved.

—*Convex optimization-based policy (Convex-opt).* The linear predictor has been designed using a 3rd order polynomial equation [Zanini et al. 2010a], an observation window of $600ms$, and a prediction length equal to $50ms$ in the future. We assumed two frequency inputs controlling the MPSoC. The first frequency input controls cores 1, 4, 5, and 8. The second input sets the frequency value for cores 2, 3, 6, and 7. We suppose that the scheduler performs a workload balancing strategy on the cores to have all the cores running with potentially the same (or very similar) active frequency. By doing this, we assume that the power consumption of the cores depends mostly on their frequency setting. To solve the problem formulation, we use CVX [Grant et al. 2012], an efficient convex optimization solver. Optimized implementations of this software take a few microseconds (for the case study described in the experimental setup section) to run on a state-of-the-art computing machine [Eldar et al. 2010; Boyd and Wegbreit 2007], which is at least three orders of magnitude less than the interval between two consequent applications of the policy (from 10ms to 100ms). The computation overhead of the online controller has no effect on the workload performance, because this overhead is really small (less than 1%) compared to the 10ms period between two consequent executions of the online controller. Moreover, any modification to the threshold temperature will also not increase the overhead of the online controller, because the number of variables remains the same in the optimization problem.

7. EXPERIMENTAL RESULTS

This section compares the four families of proposed thermal control methods. The executed workload and working temperature trade-offs are shown in Section 7.1, while Section 7.2 shows the analysis of thermal and frequency variations for each policy.

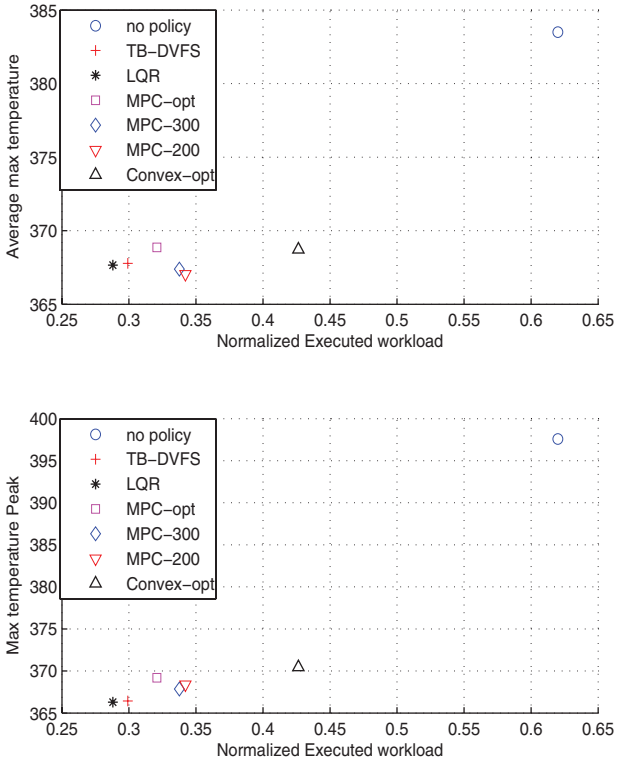


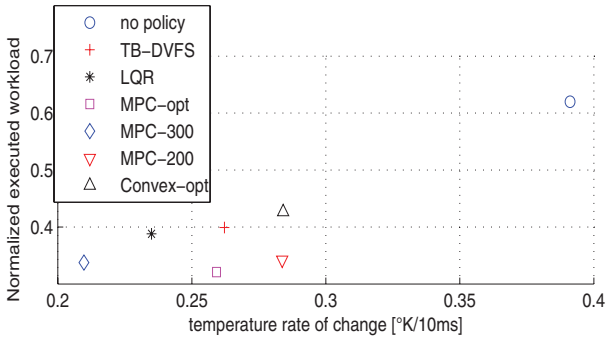
Fig. 8. No thermal control (no policy). Temperature vs. executed workload normalized to the one that can be executed by the MPSoC running with the highest possible frequency setting. The maximum temperature limit is set to $370^{\circ}K$, except in the case when no thermal control (no policy) is applied.

7.1. Executed Workload and Working Temperature

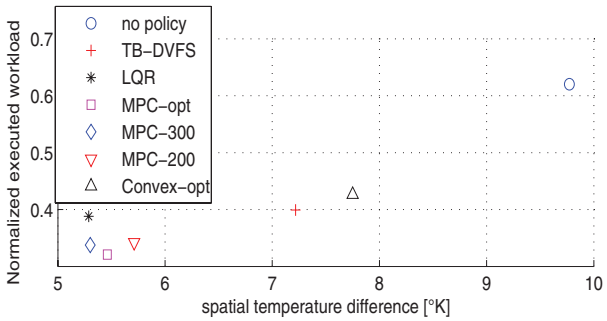
Figure 8 compares the performance of the policies by plotting the normalized executed workload versus the thermal profile. The normalized executed workload is the workload executed by the policies normalized to the one that can be executed by the MPSoC running with the highest allowable frequency setting. The top plot analyzes the average maximum chip temperature, while the bottom shows the maximum MPSoC temperature peak. The average maximum chip temperature is the average of the maximum temperature over all the MPSoC.

As these figures show, in the case that no thermal policy is used (upper-right circle), even for an average workload of less than 65%, the maximum chip temperature gets to almost $398^{o}K$. However, all the thermal policies compared in this work are able to avoid this problem. Indeed, the punctual maximum temperature peak of all the policies is less than $370.46^{o}K$. The reason for this extra $0.46^{o}K$ (above the predefined maximum temperature threshold $370^{\circ}K$) is the mismatch between the detailed thermal model used in the MPSoC thermal simulation framework [Atienza et al. 2008; Murali et al. 2008] and the model used by the thermal control strategies definition.

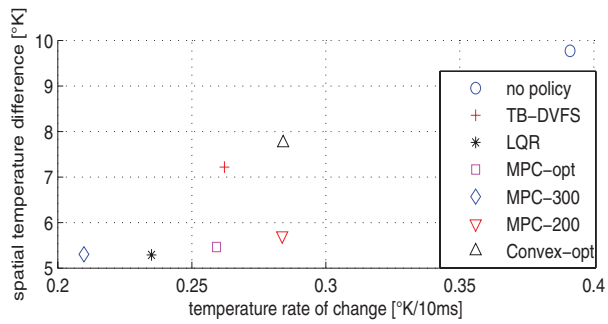
The best performance in terms of executed workload is provided by convex optimization-based policy. This policy outperforms the TB-DVFS in terms of executed workload by a factor of 50%, by having approximately the same average temperature. The approximated MPCs (with 200–300 vertices) provide almost the same performance with very small differences in temperatures. The optimum MPC provides



(a) Normalized executed workload vs. temporal temperature gradient.



(b) Normalized executed workload vs. spatial temperature gradient.



(c) Spatial vs. temporal temperature gradient.

Fig. 9. Temperature gradients analysis.

approximately 5% lower performance. The reason being that the optimum MPC changes the frequency values more often than its approximated versions. The change in the frequency setting has an overhead in terms of additional power dissipation that is not considered in the problem formulation. However, our results show that this effect leads to a loss of performance less than 6% when compared with the approximated MPC.

The TB-DVFS and the LQR policies show lower performances as they provide 50% and 25% less executed workload than the convex-optimization approach and the approximated MPCs policies, respectively. The LQR policy shows an executed workload that is a few percent lower than the same, compared with the TB-DVFS that is much simpler. Advantages of the LQR are shown in Figure 9. Temperature gradients are a concern not only in space, but also in time. The frequent abrupt change in working

frequencies and voltages produces thermal cycles that raise the failure rate of the system [Haase et al. 2006]. In addition, abrupt power-mode transitions in voltage and frequency scaling waste additional power [Jung and Pedram 2008].

Figure 9 consists of three plots. The top plot compares the policies according to the executed workload and changes of temperature. On the y-axis, it shows that the workload executed by the policies normalized to the one that can be executed by the MPSoC running with the highest allowable frequency setting. On the x-axis, it shows the temperature rate of change. This metric represents the mean of the maximum absolute temperature difference in degrees for the same MPSoC location between two consecutive applications of the policy. Thus, its dimension is ${}^{o}K/(T_{pol} = 10ms)$. The middle plot compares the policies according to their executed workload and spatial temperature difference. This metric represents the mean of the maximum absolute temperature difference in degrees Kelvin between two neighboring units. Overall, these two plots compare the policies according to the executed workload and temperature variations. The first considers variations in the time domain, while the second outlines variations in the space domain. Finally, the bottom plot of the figure analyzes the correlation between time and space temperature gradients for all the compared thermal control policies.

As the plots of Figure 9 show, both the spatial temperature difference and the temperature rate of change are reduced in all the policies by a factor up to 2x with respect to the case in which no thermal control is applied. Moreover, this is achieved by keeping the maximum temperature under the maximum threshold ($370^{\circ}K$) that ensures the reliable operation of the target eight-core MPSoC.

In all the plots of Figure 9, the convex optimization approach shows the highest variations in both the time and the space domain. The convex optimization-based approach indeed does not target the minimization of space and time temperature gradient, but it targets only the minimization of the power consumption and the difference between the workload requested and the one executed by the MPSoC. Therefore, it provides the best performance in terms of executed workload at the cost of a less uniform and a more rapidly changing thermal profile.

The policy with the smallest thermal variations (both in time and space) is the approximated MPC with 300 vertices and the LQR. In all graphs, the LQR shows a performance comparable with those of more complex techniques, like the MPC-300. The LQR, in terms of keeping the temperature under the predefined threshold, works with the same principle as the TB-DVFS. If we compare these two policies, a reduction of up to 15% and 45% in time and space temperature variations is achieved, respectively. Moreover, the LQR performs the best in minimizing spatial temperature gradients, because the LQR policy directly addresses thermal gradients as the main priority. This policy makes the thermal profile more uniform.

7.2. Thermal and Frequency Variations

The frequent abrupt change in working frequencies and voltages produces thermal cycles that raise the failure rate of the system [Haase et al. 2006]. The effect of thermal cycling on the reliability of a chip can be modeled by the Coffin-Manson relation, which relates, in an exponential way, the number of cycles to failure to the magnitude of thermal cycling [JEDEC 2003].

This section makes a study about thermal and frequency variations generated during the runtime execution of the compared thermal management policies. The more the policy acts smoothly while responding to workload demand, the more both these variations are small.

Figure 10 compares the policies according to the average frequency rate of change (x-axis) and the average temperature rate of change (y-axis). These two metrics

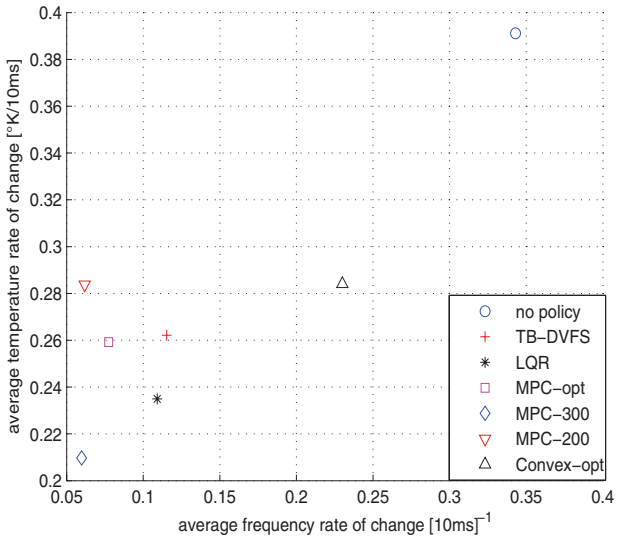


Fig. 10. Temperature rate of change vs. frequency rate of change.

represent, respectively, the average frequency and temperature variation that occur between two consequent applications of the thermal control method. In our case study, the policies are applied every 10ms, and so the metrics dimensions are [$10ms^{-1}$] and [$^{\circ}K/10ms$], respectively.

Figure 10 shows that thermal management policies are reduced up to 6x and 1.8x frequency and temperature variations, respectively, compared with the case where no thermal management policies are used. However, differences are observed in the average temperature oscillations due to the different control policies. The worst overall performance vs. temperature oscillations value is provided by the convex optimization-based approach policy. Moreover, Figure 10 outlines that the optimum MPC changes the operating frequency more often than the approximated MPC options, in order to control the temperature variations. This situation occurs because the approximated MPC policies have fewer regions, thus they change the frequency settings less often.

In addition, although the explicit MPC regulators do not consider temperature gradients in their optimization, the MPSoC shows lower gradients than the LQR policy, which directly considers the importance of gradient minimization in its problem formulation. This is due to the nonlinear saturation block that has been added to the LQR regulator to avoid the temperature exceeding the threshold temperature. In fact, this nonlinear block degrades the performance of the linear regulator in cases of heavy workload scenarios, like the one we are simulating.

8. CONCLUSIONS

In this article, we have proposed a new unified problem formulation using model predictive control (MPC) theory to explore the main features of four main thermal control algorithms for thermal management of MPSoCs. Our formulation has shown that it is possible to expose, from a theoretical viewpoint, the distance between the different control techniques regarding computation complexity, heat forecasting precision, memory requirements, adaptability to different workload conditions at runtime, and smoothness in peak temperature and thermal balancing control. Therefore, it is possible to identify the specific thermal policy that provides the necessary trade-offs to achieve the best online smooth thermal control for a specific set of constraints in a

concrete MPSoC architecture target. Finally, we have implemented the policies on an MPSoC-simulation platform and performed experiments on a model of the eight-core Niagara-1 multi-core architecture using benchmarks ranging from Web-accessing to playing multimedia. Our results have shown that the theoretical strengths and weaknesses between the compared thermal policies are effectively followed by real-life setups. Thus, the proposed unified MPC-based thermal control formulation provides MPSoC designers with a theoretical basis and analytical relations between speed, voltage, power, and temperature, as well as thermal management horizon, to select the most appropriate thermal control family for specific working constraints of each target MPSoC into its early-phase definition.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Martino Ruggiero for his help in the development of the MPSoC hardware simulation platform with the software MPC control interface.

REFERENCES

Atienza, D., Del Valle, P. G., Paci, G., Poletti, F., Benini, L., De Micheli, G., Mendias, J. M., and Hermida, R. 2008. Hw-sw emulation framework for temperature-aware design in mpsocs. *ACM Trans. Des. Autom. Electron. Syst.12,* 3, 26:1–26:26.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica 38,* 1, 3–20.

Benini, L. and De Micheli, G. 1998. *Dynamic Power Management: Design Techniques and CAD Tools*. Springer, Berlin.

Benini, L., Bogliolo, A., Paleologo, G. A., and De Micheli, G. 1999. Policy optimization for dynamic power management. *IEEE Trans. CAD Integ. Circuits Syst. 18*, 6, 813–833.

Bircher, W. L. and John, L. K. 2012. Complete system power estimation using processor performance events. *IEEE Trans. Comput. 61*, 563–577.

Bogdan, P., Jain, S., Tornero, R., and Morculescu, R. 2012. An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads. In *Proceedings of the International Symposium on Networks on Chip.* 35–42.

Borkar, S. 1999. Design challenges of technology scaling. *IEEE Micro 19*, 23–29.

Bose, P. 2003. Power-efficient microarchitectural choices at the early design stage. In *Proceedings of the Power-Aware Computer Systems Conference*. 1–6.

Boyd, S. and Vandenberge, L. 2004. *Convex Optimization*. Cambridge University Press.

Boyd, S. P. and Wegbreit, B. 2007. Fast computation of optimal contact forces. *IEEE Trans. Robotics 23*, 1117–1132.

Cochran, R. and Reda, S. 2010. Consistent runtime thermal prediction and control through workloadphase detection. In *Proceedings of the Design Automation Conference*. 62–67.

Coskun, A. K., Atienza, D., Rosing, T. S., Brunschwiler, T., and Michel, B. 2010. Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 111–116.

Coskun, A. K., Rosing, T. S., and Gross, K. C. 2008a. Proactive temperature balancing for low cost thermal management in mpsocs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 250–257.

Coskun, A. K., Rosing, T. S., and Gross, K. C. 2008b. Temperature management in multiprocessor socs using online learning. In *Proceedings of the Design Automation Conference*. 890–893.

Coskun, A. K., Rosing, T. S., and Whisnant, K. 2007. Temperature aware task scheduling in mpsocs. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 1659–1664.

Donald, J. and Martonosi, M. 2006. Techniques for multicore thermal management: Classification and new exploration. In *Proceedings of the Annual International Symposium on Computer Architecture*. 78–88.

Eldar, Y. C., Luo, Z.-Q., Ma, K., Palomar, D. P., and Sidiropoulos, N. D. 2010. Convex optimization in signal processing. *IEEE Signal Process. Mag. 27*, 19–145.

Franklin, G. F. Powell, J. D., and Workman, M. L. 1997. *Digital Control of Dynamic Systems*. McGraw Hill, New Your, NY.

GRANT, M. ET AL. 2012. Cvx: Matlab software for disciplined convex programming. http://www.stanford.edu/ boyd/cvx/.

HAASE, J., DAMM, M., HAUSER, D., AND WALDSCHMIDT, K. 2006. Reliability-aware power management of multicore processors. In *From Model-Driven Design to Resource Managenment for Distributed Embedded Systems 225*, 205–214.

HALFHILL, T. R. 2000. Transmeta breaks x86 low power barrier. *Microprocess. Rep.*

HANUMAIAH, V. AND VRUDHULA, S. 2012. Temperature-aware dvfs for hard real-time applications on multicore processors. *IEEE Trans. Comput. 61*, 1484–1494.

HUANG, W., SANKARANARAYANAN, K., SKADRON, K., RIBANDO, R. J., AND STAN, M. R. 2008. Accurate, pre-rtl temperature-aware design using a parameterized,geometric thermal model. *IEEE Trans. Comput. 57*, 1277–1288.

JEDEC. 2003. Failure mechanisms and models for semiconductor devices. In *Proceedings of the Jedec Solid State Tech. Association*.

JONES, C. N. AND MORARI, M. 2010. Polytopic approximation of explicit model predictive controllers. *IEEE Trans. Autom. Control 55,* 11, 2542–2553.

JUNG, H. AND PEDRAM, M. 2008. Continuous frequency adjustment technique based on dynamic workload prediction. In *Proceedings of the VLSI Design Conference*. 249–254.

KANG, K., KIM, J., YOO, S., AND KYUNG, C.-M. 2011. Runtime power management of 3-d multi-core architectures under peak power and temperature constraints. *IEEE Trans. Comput. Aided Des. Integ. Circuits Syst. 30,* 6, 905–918.

KONGETIRA, P., AINGARAN, K., AND OLUKOTUN, K. 2005. Niagara: A 32-way multithreaded sparc processor. *IEEE Micro 25,* 2, 21–29.

KVASNICA, M., GRIEDER, P., BAOTIC, M., AND MORARI, M. 2004. Multi-Parametric Toolbox (MPT). http://control.ee.ethz.ch/∼mpt/.

LAUB, A., HEATH, M., PAIGE, C., AND WARD, R. 1987. Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Trans. Autom. Control 32*, 115–122.

LEE, J. S., SKADRON, K., AND CHUNG, S. W. 2010. Predictive temperature-aware dvfs. *IEEE Trans. Comput. 59,* 1, 127–133.

LU, Y.-H., ŠIMUNIĆ, T., AND DE MICHELI, G. 1999. Software controlled power management. In *Proceedings of the 7th international Workshop on Hardware/Software Codesign*. 157–161.

MAGNO, M., BRUNELLI, D., THIELE, L., AND BENINI, L. 2009. Adaptive power control for solar harvesting multimodal wireless smart camera. In *Proceedings of the International Conference on Distributed Smart Cameras*. 1–6.

MERCHANT, A., MELAMED, B., SCHENFELD, E., AND SENGUPTA, B. 1996. Analysis of a control mechanism for a variable speed processor. *IEEE Trans. Comput. 45,* 7, 793–801.

MUKHERJEE, R. AND MEMIK, S. O. 2006. Physical aware frequency selection for dynamic thermal management in multi-core systems. In *Proceedings of the International Conference on Computer-Aided Design*. 547–552.

MURALI, S., MUTAPCIC, A., ATIENZA, D., GUPTA, R., BOYD, S., BENINI, L., AND DE MICHELI, G. 2008. Temperature control of high-performance multi-core platforms using convex optimization. In *Proceedings of the Conference on Design, Automation and Test in Europe*. 110–115.

OGRAS, U. Y., MARCULESCU, R., MARCULESCU, D., AND JUNG E. G. 2009. Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Trans. VLSI Syst. 17*, 3, 330–341.

QIU, Q., WU, Q., AND PEDRAM, M. 1999. Stochastic modeling of a power-managed system: construction and optimization. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 194–199.

RABAEY, J. 2009. *Low Power Design Essentials*. Springer, Berlin.

SABRY, M. M., COSKUN, A. K., AND ATIENZA, D. 2010. Fuzzy control for enforcing energy efficiency in high-performance 3d systems. In *Proceedings of the International Conference on Computer-Aided Design*. 642–648.

SABRY, M.-N. 2005. High-precision compact-thermal models. *IEEE Trans. Compon. Packag. Manuf. Technol. 28*, 623–629.

SIMUNIC, T., BOYD, S. P., AND GLYNN, P. 2004. Managing power consumption in networks on chips. *IEEE Trans. Very Large Scale Integration VLSI Syst. 12,* 1, 96–107.

SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim. 1*, 94–125.

Tondel, P., Johansen, T. A., and Bemporad, A. 2003. Evaluation of piecewise affine control via binary search tree. *Automatica 39*, 945–950.

Wang, Y., Ma, K., and Wang, X. 2009. Temperature-constrained power control for chip multiprocessors with online model estimation. In *Proceedings of the Annual International Symposium on Computer Architecture*. 314–324.

Warren, J. D., Schaefer, S., Hirani, A. N., and Desbrun, M. 2007. Barycentric coordinates for convex sets. *Adv. Comput. Math. 27*, 319–338.

Xie, Y. and Hung, W.-L. 2006. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (mpsoc) design. *J. VLSI Signal Process. Syst. 45,* 3, 177–189.

Zanini, F., Atienza, D., Benini, L., and De Micheli, G. 2009a. Multicore thermal management with model predictive control. In *Proceedings of the European Conference on Circuit Theory and Design*. 90–95.

Zanini, F., Atienza, D., De Micheli, G., and Boyd, S. P. 2010a. Online convex optimization-based algorithm for thermal management of mpsocs. In *Proceedings of the Great Lakes Symposium on VLSI*. 203–208.

Zanini, F., Atienza, D., and De Micheli, G. 2009. A control theory approach for thermal balancing of mpsoc. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 37–42.

Zanini, F., Jones, C. N., Atienza, D., and Micheli, G. D. 2010b. Multicore thermal management using approximate explicit model predictive control. In *Proceedings of the International Symposium on Circuits and Systems*. 3321–3324.

Zhang, Y. and Srivastava, A. 2010. Adaptive and autonomous thermal tracking for high performance computing systems. In *Proceedings of the Design Automation Conference*. 68–73.

Received March 2012; revised July 2012; accepted September 2012