J Electron Test (2013) 29:521–535
DOI 10.1007/s10836-013-5397-5

Efficient Worst-Case Temperature Evaluation for Thermal-Aware Assignment of Real-Time Applications on MPSoCs

Lars Schor · Iuliana Bacivarov · Hoeseok Yang · Lothar Thiele

Received: 15 September 2012 / Accepted: 15 July 2013 / Published online: 18 August 2013

Abstract The reliability of multiprocessor system-on-chips (MPSoCs) is nowadays threatened by high chip temperatures leading to long-term reliability concerns and short-term functional errors. High chip temperatures might not only cause potential deadline violations, but also increase cooling costs and leakage power. Pro-active thermal-aware allocation and scheduling techniques that avoid thermal emergencies are promising techniques to reduce the peak temperature of an MPSoC. However, calculating the peak temperature of hundreds of design alternatives during design space exploration is time-consuming, in particular for unknown input patterns and data. In this paper, we address this challenge and present a fast analytic method to calculate a non-trivial upper bound on the maximum temperature of a multi-core real-time system with non-deterministic workload. The considered thermal model is able to address various thermal effects like heat exchange between neighboring cores and temperature-dependent leakage power. Afterwards, we integrate the proposed thermal analysis method into a design-space exploration framework to optimize the task to processing component assignment. Finally, we apply the proposed method in various case studies to explore thermal hot spots and to optimize the task to processing component assignment.

Keywords Real-time systems · Compositional analysis · Thermal analysis · Design-space exploration

Responsible Editor: R. Velazco

L. Schor (✉) · I. Bacivarov · H. Yang (✉) · L. Thiele
Computer Engineering and Networks Laboratory,
ETH Zurich, 8092 Zurich, Switzerland
e-mail: lars.schor@tik.ee.ethz.ch
e-mail: hoeseok.yang@tik.ee.ethz.ch

I. Bacivarov
e-mail: bacivarov@tik.ee.ethz.ch

L. Thiele
e-mail: thiele@tik.ee.ethz.ch

1 Introduction

The increasing demand of computational performance and the better power efficiency motivates system designers to use multiprocessor system-on-chips (MPSoCs) that integrate multiple processing components, memories, and communication units on a single die. However, the use of deep submicrometer process technology to fabricate MPSoCs imposes a major rise in power densities, which in turn threats the reliability and performance of the system by inducing high chip temperatures. Thermal hot spots, i.e., areas on the chip with high temperatures, affect the design of the cooling system. In order to reduce device failures, the cooling system has to be designed for the worst-case chip temperature, i.e., the maximum chip temperature under all feasible scenarios of task arrivals [7].

Besides improving the cooling system, thermal and reliability issues have been tackled by reactive thermal management techniques or thermal-aware task allocation and scheduling algorithms. Reactive thermal management techniques such as dynamic voltage and frequency scaling (DVFS) keep the maximum temperature under a given threshold [9, 13]. However, the drawback of reactive thermal management techniques is the significant degradation of performance caused by stalling or slowing down the processor [7].

When the workload of the system is known, pro-active thermal-aware allocation and scheduling techniques that

avoid thermal emergencies and thus, a reduction in performance, might be preferable over reactive thermal management techniques. In particular, by selecting an optimal frequency, voltage, and task assignment, the peak temperature can significantly be reduced so that a certain quality of service level can be guaranteed at design-time [6, 7, 10, 17]. Nonetheless, prior work either lowers the average temperature or assumes deterministic workload where the maximum temperature of the system can be calculated by simulating the system.

However, unknown input patterns and data cause the workload to be non-deterministic so that the maximum possible chip temperature under all feasible scenarios of task arrivals is difficult to identify. Only when the corner case that actually leads to the maximum temperature of the system is considered, simulation-based thermal analysis techniques do not lead to an undesired underestimation of the maximum temperature. However, calculating this critical workload has been shown to be time-consuming [20], so that calculating the peak temperature of hundreds of design alternatives during design space exploration would be infeasible.

In this paper, we first present a fast analytic method to calculate an upper bound on the maximum temperature of an MPSoC with non-deterministic workload. The proposed method has a time-complexity that only depends on the number of processing components, which enables efficient design-space exploration. The considered thermal model is able to address various thermal effects like heat exchange between neighboring cores and temperature-dependent leakage power. We use the well-established standard event model [11] to model non-determinism in the workload, i.e., we consider periodic event streams with jitter and delay. Real-time calculus [23], a formal method for schedulability and performance analysis of real-time systems, is applied to upper bound the workload that might arrive in any time interval. Although arrival curves constrain the maximum possible workload, infinitely many traces comply with this specification. Thus, our method identifies the critical workload trace that leads to the worst-case chip temperature. The only requirement of the method is that the real-time scheduling algorithms are work-conserving, i.e., the respective processing component has to process as soon as there is an event in its ready queue. However, this applies to most of the traditional scheduling algorithms as, for example, earliest-deadline-first (EDF), rate-monotonic (RM), fixed-priority (FP), and deadline-monotonic (DM).

We then integrate the proposed thermal analysis method into a design-space exploration framework intended to optimize the task to processing component assignment at design-time, prior to deployment and execution. We formulate this optimization problem with the objective to minimize the worst-case chip temperature. Clearly, the reliability of a system is also affected by other (thermal) metrics such as, for instance, the temperature gradient. However, due to the above listed reasons, we selected the minimization of the worst-case chip temperature as the objective in our framework. Finally, we solve the optimization problem using simulated annealing and provide an extensive evaluation of its performance.

This paper is based on the work published in [19], which formally proves the thermal analysis method. Nonetheless, prior work does not integrate the thermal analysis method into an optimization framework. Thus, we extend the previous work by first formulating a thermal-aware task assignment problem, and then integrating the thermal analysis method into a framework to solve the optimization problem. In addition, this paper gives a broader coverage of related work, the proposed technique is more detailed, and a richer set of experiments is carried out. Therefore, the contributions of this paper can be summarized as follows:

- The considered system is formally described along with an overview of worst-case chip temperature analysis.
- A mathematical expression for a non-trivial upper bound on the worst-case chip temperature of a multi-core system with non-deterministic workload is derived. In contrast to previous work, the time-complexity of the proposed method only depends on the number of processing components.
- The task to processing component assignment is formulated as an optimization problem to minimize the worst-case chip temperature prior to execution.
- In various case studies, the proposed method is applied to explore the temperature distribution and optimize the task to processing component assignment.

The paper continues with a discussion of related work. Afterwards, in Section 3, the considered thermal and computational models are introduced. In Section 4, the thermal analysis methods are induced. In Section 5, the task to processing component assignment is formulated as an optimization problem. Finally, in Section 6, case studies are presented to highlight the viability of our method.

2 Related Work

As the use of dynamic thermal management (DTM) techniques yields in multi-core systems to reduced system reliability, a degradation of performance [4], and high complexity [7], various pro-active thermal-aware binding and scheduling techniques have been proposed in recent years. In [6, 7, 9, 10, 17], the maximum temperature is reduced so that the performance requirements are still met. For instance, in [17], the thermal-aware scheduling problem is formulated as a convex optimization problem. In [6], a mixed-integer linear programming formulation is stated to

solve the thermal-aware scheduling problem. In [7], a similar problem is solved for minimizing the energy consumption and reducing thermal hot spots. Finally, a global scheduling algorithm is proposed in [10] so that all cores are running at their ideally preferred speed and the peak temperature is optimally reduced. However, in all of these works, the temperature is obtained by either calculating the average temperature or the workload is assumed to be deterministic so that the maximum temperature of the system can be calculated by simulating the transient temperature evolution.

Evaluating the temperature characteristics is typically a two-step process. First, the transient power dissipation is determined by a software-based [1, 24] or hardware-based [26] power-aware simulator. Having the transient power dissipation, either the average temperature is calculated based on steady-state analysis [31] or the transient temperature evolution is obtained by simulating the system in a thermal simulator [12, 21, 22]. However, due to the complexity of today's systems, it is difficult to identify corner cases that actually lead to the maximum temperature of the system. Consequently, simulation-based thermal analysis may lead to undesired underestimations of the maximum temperature.

In this work, we use a different approach. Similar to well-known best-case / worst-case timing analysis methods for multiprocessor systems, we use formal analysis methods to predict the maximum temperature of a real-time system. For example, modular performance analysis (MPA) [29] or SymTA/S [11] provide upper and lower bounds on the latency of a system with non-deterministic workload. In particular, a first formal analysis method to calculate the worst-case chip temperature of a multi-core system with non-deterministic workload has been proposed in [20]. By incorporating the temperature into real-time analysis, real-time deadlines can be guaranteed even if reliability is subject to high temperatures. However, as the method proposed in [20] uses linear search to calculate a tight bound on the worst-case chip temperature, its evaluation time is too long for design space exploration of multi-core systems with tens of processing components. In this work, we address this challenge and describe a method to calculate an upper bound on the maximum temperature of a multi-core system with a time complexity that only depends on the number of processing components. Finally, we integrate the proposed method into a thermal-aware task assignment strategy to minimize the worst-case chip temperature subject to real-time deadlines.

3 System Models

This section introduces the formal models to analyze a real-time application on an MPSoC.

Notation Bold characters are used for vectors and matrices and non-bold characters for scalars. For example, $\mathbf{H}$ denotes a matrix whose (k, ℓ)-th element is $H_{k\ell}$ and $\mathbf{T}$ denotes a vector whose k-th element is T_k.

3.1 Computational Model

In this work, we consider an MPSoC with a set of processing components $\boldsymbol{\Theta}$. The considered computational model of each processing component Θ_ℓ is expressed using abstractions in real-time calculus [23]. In particular, we suppose that events with a total workload of $R_\ell(s,t)$ time units arrive at component Θ_ℓ in time interval $[s,t)$ and each event has a constant workload of Δ_ℓ^A time units. Thus, for any fixed s, $R(s,t)$ is a staircase function that increases its value by its computation time when an event arrives. The arrival curve α_ℓ upper bounds all possible cumulative workloads:

$$R_\ell(s,t) \leq \alpha_\ell(t-s) \quad \forall s < t \tag{1}$$

with $\alpha_\ell(0) = 0$. Figure 1a illustrates the concept of arrival curves for the widely-used standard event model where the event stream is defined by the parametric triple (p_ℓ, j_ℓ, d_ℓ) with period p_ℓ, jitter j_ℓ, and minimum inter-arrival distance between events d_ℓ [11]. For the rest of this paper, we assume that an event stream is always characterized by these three parameters.

We suppose that processing components are work-conserving. In other words, they will be in 'active' mode as long as there are events in their ready queue. The accumulated computing time $Q_\ell(s,t)$ describes the amount of time units that component Θ_ℓ is spending to process an incoming workload of $R_\ell(s,t)$ time units. It is upper bounded by $\gamma_\ell(\Delta)$ for all intervals of length $\Delta < t$ [28]:

$$Q_\ell(t-\Delta,t) \leq \gamma_\ell(\Delta) = \inf_{0\leq\lambda\leq\Delta} \{(\Delta-\lambda)+\alpha_\ell(\lambda)\}. \tag{2}$$

For any fixed s with $s < t$, the accumulated computing time $Q_\ell(s,t)$ is monotonically increasing and has either slope 1 or 0. When the slope is 1, the component is in 'active' mode, i.e., it is processing events. When the component is idle, i.e., in sleep mode, the slope is 0. Thus, we express the processing mode by the mode function:

$$S_\ell(t) = \frac{\mathrm{d}Q_\ell(s,t)}{\mathrm{d}t} \in \{0,1\}. \tag{3}$$

In Fig. 1b, the computing model is illustrated by comparing a typical cumulated workload curve, the resulting accumulated computing time, and the associated mode function. The upper bound on the computing time is characterized by the length of the first active interval b_ℓ, also called *burst*, the length Δ_ℓ^A of every other interval with slope 1, and the length Δ_ℓ^I of every interval with slope 0. While b_ℓ and Δ_ℓ^A are constant for the considered computational model, we also assume for computational simplicity that the upper

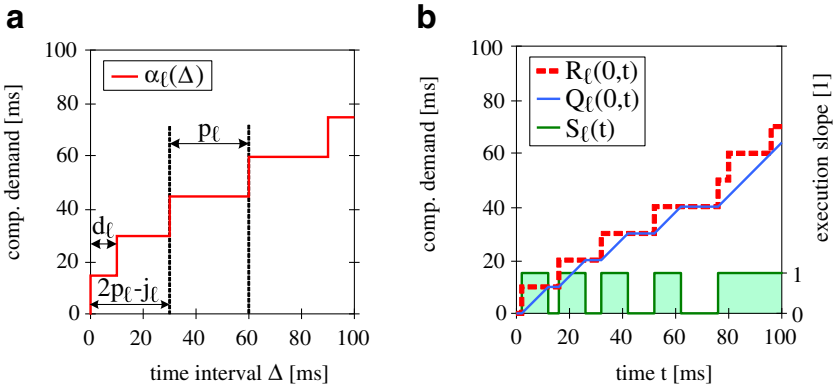

Fig. 1 Examples of typical arrival curves and accumulated computing time curves. **a** Arrival curve that corresponds to an event stream defined by the period p_ℓ, the jitter j_ℓ, and the minimum inter-arrival distance between events d_ℓ. **b** Typical cumulated workload curve $R_\ell(0,t)$ with the resulting accumulated computing time $Q_\ell(0,t)$ and the associated mode function $S_\ell(t)$

bound on the accumulated computing time is selected such that all non-increasing intervals have the same length.

3.2 Thermal Model

The considered thermal model of an MPSoC describes the temperature evolution by means of an equivalent *RC* circuit [6, 12, 20, 21]. In particular, we model the layout of the chip by four layers, namely heat sink, heat spreader, thermal interface, and silicon die. Each layer is divided into a set of blocks according to the architectural-level units. In our case, we select a processing component abstraction, i.e., we represent each processing component as an individual node with separate power source and temperature characteristics. Even though a finer granularity could have been selected, the processing component abstraction has been shown to be accurate enough for system-level optimization [8, 30]. As 12 additional nodes are introduced in the heat spreader and heat sink layers to model the area that is not covered by the subjacent layer, a multi-core system with $|\Theta|$ processing components is modeled by $n = 4 \cdot |\Theta| + 12$ nodes.

The n-dimensional temperature vector $\mathbf{T}(t)$ at time t is described by a set of first-order differential equations:

$$\mathbf{C} \cdot \frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \left(\mathbf{P}(t) + \mathbf{K} \cdot \mathbf{T}^{\mathrm{amb}}\right) - (\mathbf{G} + \mathbf{K}) \cdot \mathbf{T}(t) \tag{4}$$

with $\mathbf{C}$ the thermal capacitance matrix, $\mathbf{G}$ the thermal conductance matrix, $\mathbf{K}$ the thermal ground conductance matrix, $\mathbf{P}$ the power dissipation vector, and $\mathbf{T}^{\mathrm{amb}} = T^{\mathrm{amb}} \cdot [1, \ldots, 1]'$ the ambient temperature vector. The initial temperature vector is denoted as $\mathbf{T}^0$ and the system is assumed to start at time $t^0 = 0$.

A linear dependency of power dissipation on temperature [6, 16] is assumed due to leakage power:

$$\mathbf{P}(t) = \boldsymbol{\phi} \cdot \mathbf{T}(t) + \boldsymbol{\psi}(t) \tag{5}$$

where $\boldsymbol{\phi}$ is a diagonal matrix with constant coefficients, and $\boldsymbol{\psi}$ a vector. Finally, the state-space representation of the thermal model is expressed by:

$$\frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \mathbf{A} \cdot \mathbf{T}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{6}$$

with input vector $\mathbf{u}(t) = \boldsymbol{\psi}(t) + \mathbf{K} \cdot \mathbf{T}^{\mathrm{amb}}$, $\mathbf{A} = -\mathbf{C}^{-1} \cdot (\mathbf{G} + \mathbf{K} - \boldsymbol{\phi})$, and $\mathbf{B} = \mathbf{C}^{-1}$. As the thermal system is linear and time-invariant, the temperature of node k is:

$$T_k(t) = T_k^{\mathrm{init}}(t) + \sum_{\ell=1}^{n} T_{k,\ell}(t) \tag{7}$$

with $\mathbf{T}^{\mathrm{init}}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{T}^0$. $T_{k,\ell}(t)$ is the convolution of input u_ℓ and $H_{k\ell}$, i.e., the impulse response between nodes ℓ and k:

$$T_{k,\ell}(t) = \int_0^t H_{k\ell}(\xi) \cdot u_\ell(t - \xi) d\xi. \tag{8}$$

Input u_ℓ depends on the processing mode, i.e., the slope of the accumulated computing time of the processing component corresponding to node Θ_ℓ:

$$u_\ell(t) = S_\ell(t) \cdot u_\ell^a + (1 - S_\ell(t)) \cdot u_\ell^i \tag{9}$$

with $u_\ell^a = \psi_\ell^a + K_{\ell\ell} \cdot T^{\mathrm{amb}}$ if $S_\ell(t) = 1$ and $u_\ell^i = \psi_\ell^i + K_{\ell\ell} \cdot T^{\mathrm{amb}}$ if $S_\ell(t) = 0$. Nodes that do not correspond to a processing component have input $u_\ell(t) = u_\ell^i$. Similar to [20], we assume that $H_{k\ell}(t)$ is a non-negative unimodal function that has its maximum at time $\widetilde{t}^{\,H_{k\ell}}_{\max}$, see Fig. 2 for an illustration.

4 Thermal Analysis

In this section, we start by presenting some key results from peak temperature analysis, and then introduce a novel method to calculate a non-trivial upper bound on the maximum temperature without simulating the transient temperature evolution.

4.1 Peak Temperature Analysis

The worst-case chip temperature T_S^*, i.e., the maximum temperature of the system under all feasible scenarios of task arrivals, is the maximum possible temperature of all nodes:

$$T_S^* = \max\left(T_1^*, \ldots, T_n^*\right) \tag{10}$$

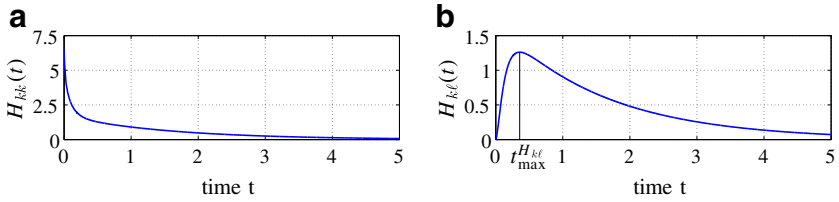

Fig. 2 Examples of typical impulse responses of the equivalent *RC* circuit to model the thermal behavior of a multi-core system. **a** Self-impulse response $H_{kk}(t)$. **b** General impulse response $H_{k\ell}(t)$

with n the number of nodes and T_k^* the worst-case peak temperature of node k. Because of non-determinism in the workload arriving at the different components, first, one has to identify the *critical* set of cumulative workload traces that leads to the worst-case peak temperature T_k^* of node k. Due to heat exchange between neighboring components, T_k^* does not only depend on the workload of component Θ_k, but also on the workload of all other components of the chip.

Once the critical set of cumulative workload traces is identified, the temperature $T_k^*(\tau)$ at a certain observation time τ is found by simulating the system with the set of critical accumulated computing times $\mathbf{Q}^{\{k\}} = \left[Q_1^{\{k\}}(0,\Delta),\ldots,Q_n^{\{k\}}(0,\Delta)\right]'$ for all $0 \le \Delta \le \tau$, where $\{k\}$ indicates that $\mathbf{Q}^{\{k\}}$ leads to the worst-case peak temperature of node k. The critical accumulated computing time $Q_\ell^{\{k\}}(0,t)$ describes the sequence of active and idle time units that, among all possible sequences of active and idle time units, leads to the maximum temperature $T_k^*(\tau)$. Thus, the remaining question is how to calculate the set of critical accumulate computing times $\mathbf{Q}^{\{k\}}$.

First, we note that the critical accumulated computing time $Q_\ell^{\{k\}}(0,\Delta)$ of node ℓ can be calculated independently of all other accumulated computing times [20]. In particular, the problem is equivalent to find the accumulated computing time $Q_\ell^{\{k\}}(0,\Delta)$ that maximizes $T_{k,\ell}$ defined as in Eq. 8. Let us define $\widetilde{t}_{\max}^{H_{k\ell}} = \tau - t_{\max}^{H_{k\ell}}$, where τ is the predefined observation time of the peak temperature, and introduce the auxiliary function v_ℓ of node ℓ, which is, starting at time t^s, one for Δ_ℓ^A time units:

$$v_\ell(t,t^s) = \begin{cases} 1 & 0 \le t^s \le t \le \min\left(t^s + \Delta_\ell^A, \tau\right) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The next theorem follows from the results of [20] and provides a method to calculate the critical accumulated computing time $\mathbf{Q}^{\{k\}}$ leading to the worst-case peak temperature T_k^* of node k.

Theorem 1 *Suppose that the accumulated computing time function* $\mathbf{Q}^{\{k\}}(0,\Delta) = \left[Q_1^{\{k\}}(0,\Delta),\ldots,Q_n^{\{k\}}(0,\Delta)\right]'$ *for all* $0 \le \Delta \le \tau$ *with* $Q_\ell^{\{k\}}(0,\Delta)$ *constructed by Algorithm 1 leads to* $T_k^*(\tau)$ *at time* τ. *When the scheduler is work-conserving,* $T_k^*(\tau)$ *is an upper bound on the highest possible value of temperature* $T_k(\tau)$ *at time* τ. *Furthermore, when* $(\mathbf{T}^\infty)^i$ *is the steady-state temperature vector if all nodes are in 'idle' mode,* $T_k^*(\tau) \ge T_k(t)$ *for all* $0 \le t \le \tau$ *and any set of feasible workload traces with the same initial temperature vector* $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$.

Algorithm 1 Calculation of the critical accumulated computing time function $Q_\ell^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$ with v_ℓ defined by Eq. 11.

Input: b_ℓ, Δ_ℓ^I, Δ_ℓ^A, p_ℓ, $\widetilde{t}_{\max}^{H_{k\ell}}$, τ, $H_{k\ell}$
Output: $Q_\ell^{\{k\}}(0,\Delta)$

1: **for all** $t^{(r)}$ in $\left[\widetilde{t}_{\max}^{H_{k\ell}}, \widetilde{t}_{\max}^{H_{k\ell}} + b_\ell - \Delta_\ell^A\right]$ **do**
   ▷ *position of the burst*
2: **for all** $t^s \in [0, \Delta_\ell^I]$ **do**
   ▷ *gap btw burst and suc. active interval*
3: $t^{(l)} = t^{(r)} - b_\ell + \Delta_\ell^A$
4: $S_\ell(t) = \begin{cases} 1 & t \in \left[t^{(r)} - b_\ell + \Delta_\ell^A, t^{(r)}\right) \\ 0 & \text{otherwise} \end{cases}$
5: **for** $i = 1$ to $\left\lceil \frac{\tau - t^{(r)}}{p_\ell} \right\rceil$ **do**
   ▷ *make trace for* $t > t^{(r)}$
6: $S_\ell(t) = S_\ell(t) + v_\ell(t, t^s + t^{(r)} + (i-1)\cdot p_\ell)$
7: **end for**
8: **for** $i = 1$ to $\left\lceil \frac{t^{(l)}}{p_\ell} \right\rceil$ ▷ *make trace for* $t < t^{(l)}$
9: $S_\ell(t) = S_\ell(t) + v_\ell(t, t^s + t^{(l)} - i\cdot p_\ell)$
10: **end for**
11: $\Upsilon = \int_0^t S_\ell(\xi)\cdot H_{k\ell}(t-\xi)\mathrm{d}\xi$
12: **if** $\Upsilon > \Upsilon^*$ **then** ▷ Υ *comparison*
13: $\Upsilon^* = \Upsilon$, $S_\ell^{\{k\}} = S_\ell$
14: **end if**
15: **end for**
16: **end for**
17: $Q_\ell^{\{k\}}(0,\Delta) = \int_0^\Delta S_\ell^{\{k\}}(\xi)\mathrm{d}\xi$ for all $0 \le \Delta \le \tau$

Algorithm 1 calculates the critical accumulated computing time $Q_\ell^{\{k\}}$ by altering both the position of burst and gap between burst and first successive active interval, see Fig. 3 for an illustration of the algorithm. Then, $Q_\ell^{\{k\}}$ is the computing time that maximizes the sum of all areas below $H_{k\ell}$ where the node is in 'active' processing mode. Afterwards, the peak temperature $T_k^*(\tau)$ of node k is obtained by simulating the system with computing time $\mathbf{Q}^{\{k\}}$.

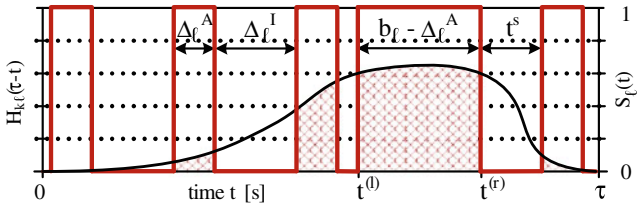

Fig. 3 Sketch of Algorithm 1 by illustrating the convolution between $H_{k\ell}(\tau - t)$ and $S_\ell(t)$

Calculating an upper bound on the maximum temperature T_S^* has time complexity $O(n^2 * m)$ with n the number of nodes. The factor m reflects the time to execute Algorithm 1 and is inversely proportional to the selected time step. Increasing the time step can improve the execution time, but might lead to a reduced accuracy.

4.2 Fast Temperature Evaluation

As indicated, Algorithm 1 might be time-consuming and hence not suited for design space exploration. Therefore, we first derive an analytical expression for the accumulated computing time that leads to a non-trivial upper bound on the peak temperature. Afterwards, we use the obtained computing time to propose a novel mathematical expression for an upper bound on the maximum temperature.

The first lemma simplifies Algorithm 1 so that it is constant time and so that the resulting upper bound on the maximum temperature at observation time τ, $\widehat{T}_k^*(\tau)$, is not smaller than the worst-case peak temperature of node k. $\widehat{\mathbf{Q}}^{\{k\}}$ denotes the critical accumulated computing time that leads to $\widehat{T}_k^*(\tau)$.

Theorem 2 *Suppose that the accumulated computing time function* $\widehat{\mathbf{Q}}^{\{k\}}(0, \Delta) = \left[\widehat{Q}_1^{\{k\}}(0, \Delta), \ldots, \widehat{Q}_n^{\{k\}}(0, \Delta)\right]'$ *for all* $0 \le \Delta \le \tau$ *with:*

$$\widehat{Q}_\ell^k(0, \Delta) = \begin{cases} \gamma\left(\widetilde{t}_{\max}^{H_{k\ell}}\right) - \gamma\left(\widetilde{t}_{\max}^{H_{k\ell}} - \Delta\right) & 0 \le \Delta < \widetilde{t}_{\max}^{H_{k\ell}} \\ \gamma\left(\widetilde{t}_{\max}^{H_{k\ell}}\right) + \gamma\left(\Delta - \widetilde{t}_{\max}^{H_{k\ell}}\right) & \widetilde{t}_{\max}^{H_{k\ell}} \le \Delta < \tau \end{cases} \tag{12}$$

leads to $\widehat{T}_k^*(\tau)$ *at time* τ. *When the scheduler is work-conserving,* $\widehat{T}_k^*(\tau)$ *is an upper bound on the highest value of temperature* $T_k(\tau)$ *at time* τ. *Furthermore, when* $(\mathbf{T}^\infty)^i$ *is the steady-state temperature vector if all nodes are in 'idle' mode,* $\widehat{T}_k^*(\tau) \ge T_k(t)$ *for all* $0 \le t \le \tau$ *and any set of feasible workload traces with the same initial temperature vector* $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$.

Proof We will prove this theorem by translating $\mathrm{Q}_\ell^{\{k\}}$ calculated by Algorithm 1 into $\widehat{\mathrm{Q}}_\ell^{\{k\}}$ calculated by Eq. 12 and show that the temperature will not decrease in every step. To this end, we observe that the temperature does not decrease if the amount of 'active' time units per time interval is either increased or shifted closer to $\widetilde{t}_{\max}^{H_{k\ell}}$ [20].

Algorithm 2 Calculation of the critical accumulated computing time function $\widehat{Q}_\ell^{\{k\}}(0, \Delta)$ for all $0 \le \Delta \le \tau$.

Input: b_ℓ, Δ_ℓ^I, Δ_ℓ^A, p_ℓ, $\widetilde{t}_{\max}^{H_{k\ell}}$, τ
Output: $\widehat{Q}_\ell^{\{k\}}(0, \Delta)$

$t^{(r)} = \widetilde{t}_{\max}^{H_{k\ell}} + b_\ell - \Delta_\ell^A$, $t^{(l)} = \widetilde{t}_{\max}^{H_{k\ell}} - b_\ell + \Delta_\ell^A$
$\widehat{S}_\ell^k(t) = \begin{cases} 1 & t \in [t^{(l)}, t^{(r)}] \\ 0 & \text{otherwise} \end{cases}$ ▷ *extended burst*
for $i = 1$ to $\left\lceil \frac{\tau - t^{(r)}}{p_\ell} \right\rceil$ **do** ▷ *trace for* $t > t^{(r)}$
  $\widehat{S}_\ell^k(t) = \widehat{S}_\ell^k(t) + v_\ell\left(t, t^{(r)} + (i-1) \cdot p_\ell\right)$
end for
for $i = 1$ to $\left\lceil \frac{t^{(l)}}{p_\ell} \right\rceil$ **do** ▷ *trace for* $t < t^{(l)}$
  $\widehat{S}_\ell^k(t) = \widehat{S}_\ell^k(t) + v_\ell\left(t, \Delta_\ell^I + t^{(l)} - i \cdot p_\ell\right)$
end for
$\widehat{Q}_\ell^k(0, \Delta) = \int_0^\Delta \widehat{S}_\ell^k(\xi) d\xi$ for all $0 \le \Delta \le \tau$

In the first step, the precedent and successive active intervals of the burst are moved to the burst so that the node is continuously active for $b_\ell + \Delta_\ell^A$ time units, compare Fig. 4a and b for an illustration. The second step makes the search for the position of the burst obsolete. To this end, the length of the burst is extended so that it covers all possible positions of the burst, i.e., $\widehat{S}_\ell(t) = 1$ for all $t \in \left[\widetilde{t}_{\max}^{H_{k\ell}} - b_\ell + \Delta_\ell^A, \widetilde{t}_{\max}^{H_{k\ell}} + b_\ell - \Delta_\ell^A\right]$, see Fig. 4c for an illustration.

Algorithm 2 is the result of these two translations. One can readily prove that in both steps, the amount of 'active' time units is either increased or shifted closer to $\widetilde{t}_{\max}^{H_{k\ell}}$. Finally, we note that Eq. 12 is equivalent to Algorithm 2, and therefore $\widehat{T}_k^*(\tau) \ge T_k^*(\tau)$. □

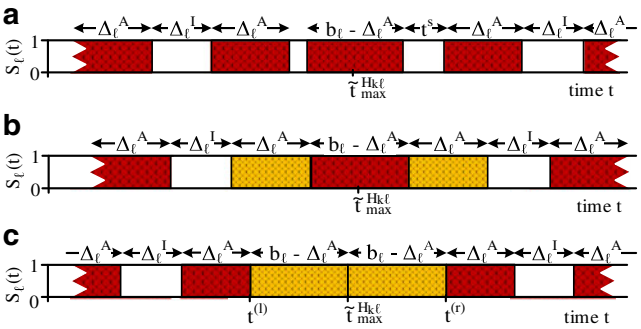

Fig. 4 Illustration of the steps to translate $Q_\ell^{\{k\}}$ into $\widehat{Q}_\ell^{\{k\}}$. **a** Starting position corresponding to the the critical accumulated computing time $Q_\ell^{\{k\}}$. **b** Translation step 1: the precedent and successive active intervals of the burst are moved to the burst. **c** Translation step 2: the length of the burst is extended so that it covers all possible positions of the burst

As Algorithm 2 is constant time, computing an upper bound on the maximum temperature of an MPSoC has time complexity $O(n^2)$ with n the number of nodes. As illustrated in Fig. 4c, the system continuously switches between active and idle mode except during the burst. Next, we show that calculating the peak temperature can further be simplified by running the processing component with constant slope $\delta_\ell = \frac{\Delta_\ell^A}{\Delta_\ell^A+\Delta_\ell^I}$ for all time units except during the burst. This lemma provides the foundation for the main theorem of this section.

Lemma 1 *Suppose that the mode function:*

$$\breve{S}_\ell(t) = \begin{cases} 1 & \widetilde{t}^{H_{k\ell}}_{\max} - b_\ell \le t \le \widetilde{t}^{H_{k\ell}}_{\max} + b_\ell \\ \delta_\ell & \textit{otherwise} \end{cases} \tag{13}$$

with utilization $\delta_\ell = \frac{\Delta_\ell^A}{\Delta_\ell^A+\Delta_\ell^I}$ *leads to* $\breve{T}^*_{k,\ell}(\tau)$ *at time* τ. *When the scheduler is work-conserving,* $\breve{T}^*_{k,\ell}(\tau)$ *is an upper bound on the highest value of* $T_{k,\ell}(\tau)$ *at time* τ.

Proof Rewriting Eq. 8 with Eq. 9 leads to:

$$T_{k,\ell}(t) = u_\ell^i \cdot \int_0^t H_{k\ell}(t-\xi)\mathrm{d}\xi + \left(u_\ell^a - u_\ell^i\right) \cdot \int_0^t S_\ell(\xi) \cdot H_{k\ell}(t-\xi)\mathrm{d}\xi.$$

As we know from Theorem 2 that $\widehat{S}(t) = \frac{\mathrm{d}\widehat{\mathcal{Q}}_\ell^{\{k\}}(0,\Delta)}{\mathrm{d}t}$ leads to $\widehat{T}_{k,\ell}(\tau)$ with $\widehat{T}_{k,\ell}(\tau) \ge T_{k,\ell}(\tau)$, we have to show that $\breve{T}_{k,\ell}(\tau) \ge \widehat{T}_{k,\ell}(\tau)$:

$$\begin{aligned}
&\left(\breve{T}_{k,\ell}(\tau) - \widehat{T}_{k,\ell}(\tau)\right) / \left(u_\ell^a - u_\ell^i\right) \\
&= \int_0^\tau \breve{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi - \int_0^\tau \widehat{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&= \int_0^{\theta^{(l)}} \breve{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi - \int_0^{\theta^{(l)}} \widehat{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&\quad + \int_{\theta^{(r)}}^\tau \breve{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi - \int_{\theta^{(r)}}^\tau \widehat{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi
\end{aligned}$$

where we used that $\breve{S}_\ell(t) = \widehat{S}_\ell(t) = 1$ for $t \in [\theta^{(l)}, \theta^{(r)}]$ with $\theta^{(l)} = \widetilde{t}^{H_{k\ell}}_{\max} - b_\ell$ and $\theta^{(r)} = \widetilde{t}^{H_{k\ell}}_{\max} + b_\ell$. By rewriting the integral from 0 to $\theta^{(l)}$ as a sum, we get:

$$\begin{aligned}
&\int_0^{\theta^{(l)}} \left(\breve{S}_\ell(\xi) - \widehat{S}_\ell(\xi)\right) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&= \sum_{i=0}^{\rho} \left(\int_{\theta^{(l)}-(i+1)\cdot p_\ell}^{\theta^{(l)}-i\cdot p_\ell} \left(\breve{S}_\ell(\xi) - \widehat{S}_\ell(\xi)\right) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \right)
\end{aligned}$$

where ρ is an integer that is selected so that $p_\ell \cdot (\rho - 1) \le \theta^{(l)} \le p_\ell \cdot \rho$. In particular, $\widehat{S}(t) = 0$ for $t \in \left[\theta^{(l)} - i \cdot p_\ell - \Delta_\ell^I, \theta^{(l)} - i \cdot p_\ell\right]$, $\widehat{S}(t) = 1$ for $t \in \left[\theta^{(l)} - (i+1) \cdot p_\ell, \theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A\right]$, and $\theta^{(l)} - i \cdot p_\ell - \Delta_\ell^I = \theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A$, see Fig. 5 for an illustration. Then, we find:

$$\begin{aligned}
&\int_{\theta^{(l)}-(i+1)\cdot p_\ell}^{\theta^{(l)}-i\cdot p_\ell} \left(\breve{S}_\ell(\xi) - \widehat{S}_\ell(\xi)\right) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&= \delta_\ell \cdot \int_{\theta^{(l)}-(i+1)\cdot p_\ell+\Delta_\ell^A}^{\theta^{(l)}-i\cdot p_\ell} H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&\quad -(1-\delta_\ell) \cdot \int_{\theta^{(l)}-(i+1)\cdot p_\ell}^{\theta^{(l)}-i\cdot p_\ell-\Delta_\ell^I} H_{k\ell}(\tau-\xi)\mathrm{d}\xi
\end{aligned}$$

where we subtracted the two integrals in the interval $\left[\theta^{(l)} - (i+1) \cdot p_\ell, \theta^{(l)} - i \cdot p_\ell - \Delta_\ell^I\right]$. Next, we lower bound the value between $\theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A$ and $\theta^{(l)} - i \cdot p_\ell$ by means of $H_{k\ell}\left(\tau - \left(\theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A\right)\right)$, and upper bound the value between $\theta^{(l)} - (i+1) \cdot p_\ell$ and $\theta^{(l)} - i \cdot p_\ell - \Delta_\ell^I$ by means of $H_{k\ell}\left(\tau - \left(\theta^{(l)} - i \cdot p_\ell - \Delta_\ell^I\right)\right) = H_{k\ell}\left(\tau - \left(\theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A\right)\right)$, as well:

$$\begin{aligned}
&\int_{\theta^{(l)}-(i+1)\cdot p_\ell}^{\theta^{(l)}-i\cdot p_\ell} \left(\breve{S}_\ell(\xi) - \widehat{S}_\ell(\xi)\right) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \\
&\ge \delta_\ell \cdot \Delta_\ell^I \cdot H_{k\ell}\left(\tau - \left(\theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A\right)\right) \\
&\quad -(1-\delta_\ell) \cdot \Delta_\ell^A \cdot H_{k\ell}\left(\tau - \left(\theta^{(l)} - (i+1) \cdot p_\ell + \Delta_\ell^A\right)\right) = 0
\end{aligned}$$

where we used the fact that $\delta_\ell = \frac{\Delta_\ell^A}{\Delta_\ell^A+\Delta_\ell^I}$ and $\delta_\ell \cdot \Delta_\ell^I - (1-\delta_\ell) \cdot \Delta_\ell^A = 0$. Similarly, we can show that:

$$\int_{\theta^{(r)}}^\tau \breve{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi - \int_{\theta^{(r)}}^\tau \widehat{S}_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\mathrm{d}\xi \ge 0$$

and therefore, $\breve{T}_{k,\ell}(\tau) - \widehat{T}_{k,\ell}(\tau) / \left(\left(u_\ell^a - u_\ell^i\right)\right) \ge 0$. □

Based on Lemma 1, we will present the main result of this section. The following theorem provides a mathematical expression to calculate a non-trivial upper bound on the maximum temperature $\breve{T}^*_k(\tau)$ of node k. Finally, according to Eq. 10, an upper bound on the maximum temperature of the system can be obtained by calculating the maximum of all individual upper bounds.

Theorem 3 *Suppose that* $T_k(t)$ *is the temperature of node* k *at time instant* t *for a set of workload functions* $\mathbf{R}(s,t)$ *that are bounded by the set of arrival curves* $\boldsymbol{\alpha}$. *When the scheduler is work-conserving, the following statements hold:*

– *The temperature:*

$$\breve{T}^*_k(\tau) = T_k^{init}(\tau) + \sum_{\ell=1}^{n} \breve{T}^*_{k,\ell}(\tau) \tag{14}$$

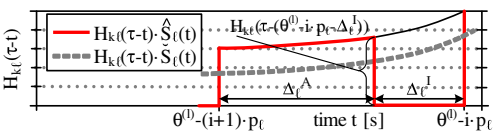

Fig. 5 Illustration of the proof of Lemma 1. The impulse response function $H_{k\ell}(t)$ is plotted for the interval $[\theta^{(l)} - (i+1) \cdot p_\ell, \theta^{(l)} - i \cdot p_\ell]$.

with

$$\breve{T}^*_{k,\ell}(\tau) = \left(u^i_\ell + \delta_\ell \cdot \left(u^a_\ell - u^i_\ell\right)\right) \cdot \int_0^\tau H_{k\ell}(t-\xi)d\xi + \left(u^a_\ell - u^i_\ell\right) \cdot (1-\delta_\ell) \cdot \int_{\tilde{t}^{H_{k\ell}}_{\max}-b_\ell}^{\tilde{t}^{H_{k\ell}}_{\max}+b_\ell} H_{k\ell}(t-\xi)d\xi \quad (15)$$

and $\delta_\ell = \frac{\Delta^A_\ell}{\Delta^A_\ell+\Delta^I_\ell}$ *is an upper bound on the highest temperature of node k at time* τ, *i.e.*, $\breve{T}^*_k(\tau) \geq T_k(\tau)$.

– *In addition, if* $(\mathbf{T}^\infty)^i$ *is the steady-state temperature vector if all nodes are in 'idle' mode,* $\widehat{T}^*_k(\tau) \geq T_k(t)$ *for all* $0 \leq t \leq \tau$ *and any set of feasible workload traces with the same initial temperature vector* $\mathbf{T}^0 \leq (\mathbf{T}^\infty)^i$.

Proof First, we rewrite Eq. 8 with Eq. 13 to derive Eq. 15. As Lemma 1 states that $\breve{T}^*_{k,\ell}(\tau) \geq \breve{T}_{k,\ell}(\tau)$ for all ℓ, and $T_k(t) = T^{init}_k(t) + \sum_{\ell=1}^n T_{k,\ell}(t)$, we get $\breve{T}^*_k(\tau) \geq T_k(\tau)$.

The second item is a simple consequence of Theorem 2. As $\breve{T}^*_k(\tau) \geq \widehat{T}^*_k(\tau)$, $\breve{T}^*_k(\tau) \geq T_k(t)$ for all $t \leq \tau$. □

Three different methods to calculate an upper bound on the maximum temperature have been presented in this section. The first method calculates the critical accumulated computing time by Algorithm 1 leading to the worst-case peak temperature T^*_k of node k. The second method calculates the accumulated computing time according to Eq. 12 leading to $\widehat{T}^*_k$, and the last method calculates an upper bound on the maximum temperature $\breve{T}^*_k$ of node k by the mathematical expression defined by Eq. 14. The relation between these three different bounds on the maximum temperature is as follows:

$$\breve{T}^*_k(\tau) \geq \widehat{T}^*_k(\tau) \geq T^*_k(\tau). \quad (16)$$

5 Minimizing the Peak Temperature

So far, we have seen a method to calculate the worst-case chip temperature, i.e, the maximum chip temperature under all feasible scenarios of task arrivals. Next, we apply this method to calculate an optimal task assignment that minimizes the worst-case chip temperature and guarantees that all real-time deadlines are met. By offering safe bounds, the resulting framework is intended to optimize the task assignment at design-time, i.e, prior to execution.

5.1 Task Model

In our task model, we assume to have a set of tasks $\boldsymbol{\nu}$ that are concurrently executed. Each task ν_j is modeled as a stream of events and has an event arrival curve $e_{\nu_j}(\Delta)$ that upper bounds the cumulative number of events arriving in any time interval of length $\Delta \geq 0$. An event has to complete its execution within D_{ν_j} time units after its arrival and function $\phi(\nu_j, \Theta_\ell)$ assigns each task ν_j the number of time units that are required to process an event on processing component Θ_ℓ. Finally, function $\Gamma(\nu_j, \Theta_\ell)$ is 1 if a task ν_j is assigned to processing component Θ_ℓ and 0 otherwise:

$$\Gamma(\nu_j, \Theta_\ell) = \begin{cases} 1 & \text{if } \nu_j \text{ executes on component } \Theta_\ell \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

Thus, the total accumulated workload of component Θ_ℓ is, in any time interval of length $\Delta \geq 0$, upper bounded by the arrival curve α_ℓ [23]:

$$\alpha_\ell(\Delta) = \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot e_{\nu_j}(\Delta) \cdot \phi(\nu_j, \Theta_\ell). \quad (18)$$

To check schedulability, we use the concept of a demand bound function [2] that models the maximum resource demand of a task. The demand bound function $\mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta)$ of task ν_j upper bounds the maximum accumulated computational demand of all events that arrive and have deadline in any interval of length Δ on processing component Θ_ℓ. Formally, the demand bound function $\mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta)$ is defined as:

$$\mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta) = e_{\nu_j}(\Delta - D_{\nu_j}) \cdot \phi(\nu_j, \Theta_\ell) \quad \forall \Delta \geq 0. \quad (19)$$

The demand bound function $\mathrm{dbf}_\ell(\Delta)$ of a processing component Θ_ℓ depends on the scheduling algorithm. For example, when an EDF scheduler is used to arbitrate between events of different tasks assigned to the same processing component, the demand bound function $\mathrm{dbf}_\ell(\Delta)$ is:

$$\mathrm{dbf}_\ell(\Delta) = \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot \mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta). \quad (20)$$

5.2 Optimization Problem

Once we have specified the task model, we can formulate the considered optimization problem:

> Given are a set of tasks $\boldsymbol{\nu}$ that are mapped onto an MPSoC with processing components $\boldsymbol{\Theta}$. Then, the goal is to select a static assignment of tasks to processing components such that all deadlines are met and the worst-case chip temperature T^*_S is minimized.

In other words, the objective of the optimization problem is to reduce the worst-case chip temperature:

$$\text{minimize } T^*_S = \max\left(T^*_1, \ldots, T^*_n\right) \quad (21)$$

where T^*_k is defined as in Eq. 14 and n is the number of nodes of the equivalent thermal *RC* circuit.

We call a processing component Θ_ℓ schedulable if the real-time deadlines of all events are met. We have to guarantee that the cumulated number of available computing

resources is in no time interval Δ smaller than the maximum resource demand, defined by the demand bound function $\mathrm{dbf}_\ell(\Delta)$. Thus, the schedulability test is written as:

$$\mathrm{dbf}_\ell(\Delta) \leq \Delta \quad \forall \Delta \geq 0 \text{ and } \nu_j \in \boldsymbol{\nu}. \tag{22}$$

Practically, the RTC toolbox [27] can be used to verify schedulability. Finally, we have to make sure that each task is assigned to only one processing component:

$$\sum_{\ell=1}^{|\boldsymbol{\Theta}|} \Gamma(\nu_j, \Theta_\ell) = 1 \quad \forall \nu_j \in \boldsymbol{\nu}. \tag{23}$$

5.3 Temperature Reduction by Voltage Scaling

The worst-case chip temperature can further be reduced by assigning each processing component its optimal frequency, i.e., the minimum operation frequency so that no real-time deadlines are missed. In the following, we extend the system model and the thermal analysis model, and formulate the optimization problem to make use of voltage and frequency scaling to reduce the power consumption, and thus, the worst-case chip temperature.

Each processing component Θ_ℓ has its own clock domain and executes at a static frequency f_ℓ with $0 \leq f_\ell \leq f_\ell^{\max}$. We suppose that the number of time units $\phi(\nu_j, \Theta_\ell)$ that an event of task ν_j has to execute on processing component Θ_ℓ scales linearly with the operation frequency. Thus, the total accumulated workload of Θ_ℓ is upper bounded by the arrival curve:

$$\alpha_\ell(\Delta) = \frac{f_\ell^{\max}}{f_\ell} \cdot \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot e_{\nu_j}(\Delta) \cdot \phi(\nu_j, \Theta_\ell). \tag{24}$$

Furthermore, we assume that the dynamic power consumption of component Θ_ℓ growths quadratically with its supply voltage v_ℓ and linearly with its operation frequency f_ℓ [18]:

$$P_{\ell,\mathrm{dyn}}(t) \propto v_\ell^2 \cdot f_\ell \cdot S_\ell(t) \tag{25}$$

Similar to [17], we suppose that the square of the supply voltage scales linearly with the operation frequency even though the results of the paper also hold for any other monotonic relation between supply voltage and frequency. Now, we can write the total power consumption as:

$$\mathbf{P}(t) = \boldsymbol{\phi} \cdot \mathbf{T}(t) + \boldsymbol{\rho} \cdot \ \mathrm{diag}(\mathbf{f})^3 \cdot \mathbf{S}(t) + \boldsymbol{\omega} \tag{26}$$

with diagonal matrix $\mathrm{diag}(\mathbf{f})$ of vector $\mathbf{f}$ and constant diagonal matrices $\boldsymbol{\rho}$ and $\boldsymbol{\omega}$. As the operation frequency is statically assigned at design-time, the thermal analysis method proposed in Eq. 4 still provides an upper bound on the maximum temperature.

In order to calculate the minimum operation frequency so that no real-time deadlines are missed, we rewrite the demand bound function $\mathrm{dbf}_\ell(\Delta)$ with the scaled operation frequency f_ℓ:

$$\mathrm{dbf}_\ell(\Delta) = \frac{f_\ell^{\max}}{f_\ell} \cdot \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot \mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta). \tag{27}$$

Finally, rewriting Eq. 22 with the above expression for the demand bound function results in the following expression for the minimum operation frequency for a processing component that uses an EDF scheduler to arbitrate between events of different tasks:

$$f_\ell = \sup_{\Delta \geq 0} \left\{ f_\ell^{\max} \cdot \frac{\sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot \mathrm{dbf}_{\nu_j,\Theta_\ell}(\Delta)}{\Delta} \right\}. \tag{28}$$

6 Case Studies

In order to evaluate the proposed analysis methods, we extended the modular performance analysis (MPA) framework [5] with the ability to calculate the maximum temperature by the discussed algorithms. Afterwards, we solve the mapping problem proposed in Eq. 5 for various task sets to illustrate the capability of the thermal analysis method.

6.1 Experimental Setup and System Description

We consider a homogeneous multi-core ARM platform with a variable number of processing components. Fixed priority preemptive scheduling is used on all processing components while a TDMA policy is employed on the shared bus that connects all processing components. Intermediate streams that cannot be represented by a period, a jitter, and a minimum interarrival distance are upper-bounded by the method presented in [15] and observation time τ is set to five seconds.

Temperature-dependency of leakage power is addressed by linearizing the model described in [21]. Table 1 summaries the parameters of the considered power model defined by Eq. 5. As we consider a homogeneous platform, every component has the same power values. HotSpot [12] is used to calculate the thermal parameters of the platform, i.e., the **C**, **G**, and **K** matrices, see Table 2 for the detailed thermal configuration. In all experiments, the traces start from the steady-state temperature in 'idle' mode,

Table 1 Power dissipation parameters of node ℓ

Parameter	Symbol	Value
Slope of power [W/K]	$\phi_{\ell\ell}$	0.023
Constant power in 'active' mode [W]	ψ_ℓ^a	8.684
Constant power in 'idle' mode [W]	ψ_ℓ^i	−5.512

Table 2 Thermal configuration of HotSpot

Parameter	Symbol	Value
Silicon thermal conductance [W/(m · K)]	k_{chip}	150
Silicon specific heat [J/(m^3 · K)]	p_{chip}	$1.75 \cdot 10^6$
Thickness of the chip [mm]	t_{chip}	3.5
Convection resistance [K/W]	r_{convec}	2
Heatsink thickness [mm]	t_{sink}	0.01
Heatsink thermal cond. [W/(m · K)]	k_{sink}	400
Heatsink specific heat [J/(m^3 · K)]	p_{sink}	$3.55 \cdot 10^6$
Ambient temperature [K]	T_{amb}	300

i.e., $\mathbf{T}^0 = (\mathbf{T}^\infty)^i$. All experiments have been performed on an Intel Core i7-2720 QM processor with 8 GB of RAM.

6.2 Worst-Case Chip Temperature Evaluation

First, we consider four benchmark applications to evaluate the performance of the proposed thermal analysis method. In particular, we compare the accuracy and the evaluation time of the novel method with the method proposed in [20].

6.2.1 Application Description

A producer-consumer (P-C), a distributed matrix multiplication, a Fast-Fourier transform (FFT), and a motion JPEG (MJPEG) decoder application are considered in the first case study. To improve the performance, the benchmark applications are split into several tasks that might run in parallel. Each task is characterized by its best-case and worst-case execution demand, which have been determined by simulating the benchmark application on the MPARM virtual platform [3]. In particular, the P-C application is split into five tasks, the matrix multiplication application into ten tasks, and the FFT application into twelve tasks. The MJPEG decoder application is split into a variable number of tasks to concurrently decode individual frames. In addition, the MJPEG decoder consists of a task to split up the input sequence into individual frames and to send the frames to the decompressing tasks. Finally, another task merges the decoded frames back into a stream.

6.2.2 Efficiency and Accuracy

First, we use an MJPEG decoder that consists of five tasks running on three processing components, thus the thermal model has order 24. The application is driven by an input stream with a periodic invocation interval of 450 ms and a jitter of 600 ms.

To evaluate our method, both the time to calculate an upper bound on the maximum temperature and the quality of this bound are analyzed. To this end, we first calculate three different upper bounds on the maximum temperature:

1. The critical accumulated computing time is computed with Algorithm 1 leading to the worst-case chip temperature T_S^*.
2. The upper bound $\widehat{T}_S^*$ is calculated by simulating the system with the critical accumulated computing time defined by Eq. 12.
3. $\breve{T}_S^*$ is calculated according to Eq. 14.

We compare T_S^*, $\widehat{T}_S^*$ and $\breve{T}_S^*$ as well as the durations to calculate the bounds. Peak temperatures and durations to calculate these temperatures are listed in Table 3 for three different mapping configurations. The time increment of Algorithm 1 is set to 1 ms. $(\mathbf{T}^\infty)^a$ and $(\mathbf{T}^\infty)^i$ are the steady-state temperature vectors if all nodes are in 'active' and 'idle' mode, respectively.

Calculating $\breve{T}_S^*$ is on average 549 times faster than calculating T_S^*, but note that the execution time of Algorithm 1 depends on the selected time increment. Furthermore, the execution time depends on the actual mapping as shown in Table 3a. We quantify the accuracy of $\breve{T}_S^*$ by means of the worst-case chip temperature T_S^*. To this end, we introduce

Table 3 Efficiency and accuracy comparison for the MJPEG decoder application

	Mapping 1	Mapping 2	Mapping 3
(a) Duration of the compared temperature analysis methods			
Duration to calculate T_S^*	35.9 s	35.0 s	40.5 s
Duration to calculate $\widehat{T}_S^*$	0.61 s	0.58 s	2.00 s
Duration to calculate $\breve{T}_S^*$	0.06 s	0.04 s	0.10 s
(b) Maximum temperature comparison for different analysis methods			
T_S^*	352.921 K	350.634 K	366.180 K
$\widehat{T}_S^*$	352.935 K	350.649 K	366.909 K
$\breve{T}_S^*$	352.937 K	350.651 K	366.910 K
$\max(\mathbf{T}^\infty)^a$	424.782 K	424.782 K	424.782 K
$\max(\mathbf{T}^\infty)^i$	310.714 K	310.714 K	310.714 K

the following notation of a relative error, which measures the normalized distance between $\breve{T}_S^*$ and T_S^*:

$$\text{error} = \frac{\breve{T}_S^* - T_S^*}{\max\,(\mathbf{T}^\infty)^a - \max\,(\mathbf{T}^\infty)^i}. \tag{29}$$

Applying this formula, the average error of our results is found to be only 0.22 %. This confirms our approach to upper bound the peak temperature by Eq. 14 instead of using Algorithm 1 to calculate the critical accumulated computing time and then simulating the system with the critical computing time. Overall, calculating $\breve{T}_S^*$ instead of T_S^* is desirable in the design flow of real-time systems as the three order of magnitude reduction in evaluation time enables a faster and more exhaustive design space exploration.

The small error is mainly attributed to the fact that the heat transfer among neighboring nodes is smaller than the self-heating. For self-heating, Eq. 12 and Algorithm 1 calculate the same critical accumulated computing time as $\widetilde{\tau}^{H_{k\ell}}_{\max}$ is equal to the observation time τ.

Finally, we compare T_S^* and $\breve{T}_S^*$ as well as the durations to calculate the bounds for the P-C, the matrix multiplication, and the FFT application. The results are listed in Table 4 for one mapping per application. They exhibit similar trends as observed with the MJPEG decoder application.

6.2.3 Comparison with a Cycle-Accurate Simulation

The method proposed in [20] has been extensively evaluated against a cycle-accurate simulation tool-chain. For completeness, we summarize the results of this evaluation; see [25] for additional details. The tool-chain is based on MPARM [3] and HotSpot [12] and key results of the evaluation are listed in Table 5 whereby the reported values are the average of six different mapping configurations. Simulating the temperature evolution on the tool-chain is on average 266 times slower than calculating the peak temperature with the analytic method proposed in [20]. The maximum chip temperature T_S^* is on average 4.8 K higher than the maximum temperature of the cycle-accurate simulation. One of the reasons for the difference is that the maximum temperature of the cycle-accurate simulation underestimates the worst-case chip temperature due to the infeasibility of an exhaustive simulation of all system configurations.

Table 4 Efficiency and accuracy comparison for the P-C, the matrix multiplication, and the FFT application

	P-C	Matrix	FFT
Duration to calc. T_S^*	25.71 s	28.02 s	26.2 s
Duration to calc. $\breve{T}_S^*$	0.09 s	0.12 s	0.15 s
T_S^*	345.402 K	354.116 K	338.832 K
$\breve{T}_S^*$	346.045 K	355.570 K	339.711 K
Error	0.56 %	1.28 %	0.77 %

Table 5 Comparison of the worst-case chip temperature and the maximum temperature observed on a cycle-accurate simulation tool chain

	P-C	Matrix	FFT	MJPEG
Exec. time reduction	293x	258x	276x	238x
T_S^*	355.4 K	355.1 K	342.6 K	362.9 K
$max(T)$ from sim.	350.3 K	351.2 K	338.8 K	356.6 K

6.2.4 Temperature Distribution on a 25-Core Processor

Next, we consider a multi-core system with 25 processing components executing an MJPEG decoder with 10 tasks. The processing components are arranged in a grid with five rows and the corresponding thermal model has order 112. We will show that the temperature distribution, and thereby the worst-case chip temperature of the system, is affected by the assignment of tasks to processing components.

Figure 6 shows the worst-case chip temperature distribution of the system for four different mappings. In Fig. 6a, the tasks are mapped onto components situated in the left top corner of the chip. Next, in Fig. 6b, the tasks are distributed among components in all four corners. In Fig. 6c, the tasks are distributed all over the chip, and finally, in Fig. 6d, the tasks are only mapped onto components in the middle of the chip. The highest peak temperature occurs in Fig. 6a and the lowest one in Fig. 6c. The difference between their worst-case chip temperatures is of about 16 K. This shows that the worst-case chip temperature can be reduced by spreading the workload over the chip. In this case, intermediate processing components with no workload act like a passive cooling system and keep hot spots separated.

6.3 Thermal-Aware Task Assignments

In the second case study, we apply the proposed temperature analysis method to calculate an optimal task assignment that minimizes the worst-case chip temperature and guarantees that all real-time deadlines are met.

6.3.1 System Description

We are still targeting the homogeneous multi-core ARM platform. The platform has two different modes to control the operation frequency. Either all processing components have a common clock domain or each processing component is supposed to have its own clock domain. The maximum operation frequency is supposed to be 1.6 GHz and the power model shown in Table 1 has been extended according

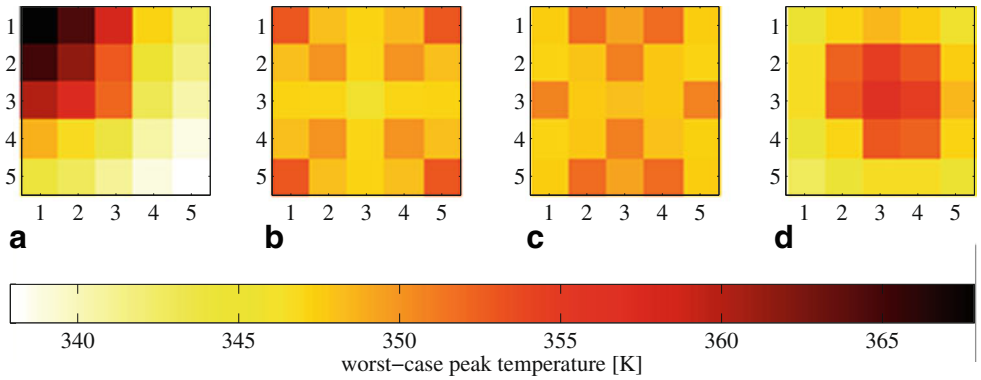

Fig. 6 Worst-case peak temperature distribution for a 25-core processor when executing an MJPEG decoder application. The processing components are arranged in a grid with five rows. **a** Mapping 1. **b** Mapping 2. **c** Mapping 3. **d** Mapping 4

to Eq. 26. An EDF scheduler is running on each processing component to arbitrate between events of different tasks assigned to the same component.

6.3.2 Simulated Annealing to Optimize the Temperature

We first suppose that each processing component is running at its maximum operation frequency, i.e., 1.6 GHz. The thermal optimization problem stated in Eq. 5 can be solved exhaustively for small task sets and platforms with a low number of processing components. Thus, we first compare the performance of a heuristic solver with the optimal solution found by exhaustively exploring the design space. The heuristic solver uses simulated annealing [14] to solve the thermal optimization problem. In addition, for comparison with the optimized task assignments, the average peak temperature of 20 feasible, i.e., schedulable random task assignments is calculated.

We consider three different hardware platforms with three, four, and six cores, respectively. Each task set is randomly generated so that the number of tasks in one set is between four and six tasks. Each task v_j is characterized by a period p_{v_j}, a jitter j_{v_j}, and a computing demand c_{v_j}. The period p_{v_j} is uniformly chosen from [1,400] *ms*, the jitter j_{v_j} is uniformly chosen from [1 ms, $2 \cdot p_{v_j}$], and the computational demand is uniformly chosen from [1, $p_{v_j} \cdot f^{\max}/5$] cycles with $f^{\max} = 1.6$ GHz. Finally, the real-time deadline of an event is set to the period of its task.

Figure 7 compares the performance of the three solvers. Exhaustively exploring the design space results in a task assignment that has a worst-case chip temperature, which is, on average, only 0.37 K smaller than the maximum temperature of the task assignment found by simulated annealing. For comparison, the average peak temperature of the random assignments is on average 3.6 K higher than the minimum peak temperature. Calculating the optimal solution for the hardware platform with six cores took on average 94.5 min and simulated annealing finished on average in 33.8 s.

6.3.3 Voltage and Frequency Scaling

Finally, we evaluate the effect of frequency and voltage scaling on the worst-case chip temperature. For a given task set, we solve the optimization problem for the following three configurations:

1. *maximum frequency*: each processing component is running at its maximum frequency.
2. *single clock domain*: the platform has a single clock domain for all processing components and is running at the minimum operation frequency so that no real-time deadline is missed.
3. *separate clock domain*: each processing component has an own clock domain and is running at the minimum operation frequency so that no real-time deadline is missed.

In other words, in the third configuration, each core has a separate frequency that is individually calculated by Eq. 28. In the second configuration, all cores are running at the same frequency and this frequency is set to the maximum frequency of all frequencies used for the third configuration.

The layout of the considered platforms is 3×1, 3×2, 3×3, and 4×4 with 3, 6, 9, and 16 cores, respectively. We compare eight different task sets per platform and each task set is randomly generated so that the number of tasks in a set is between one and three times the number of process-

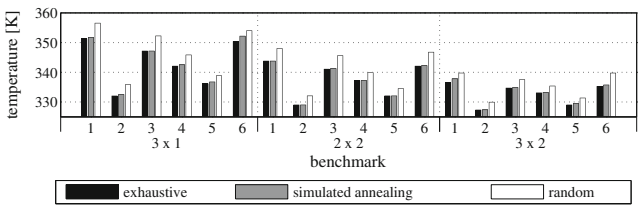

Fig. 7 Performance of different solvers for the temperature optimization problem. Three different hardware platforms with a 3×1, 2×2, and 3×2 layout are considered

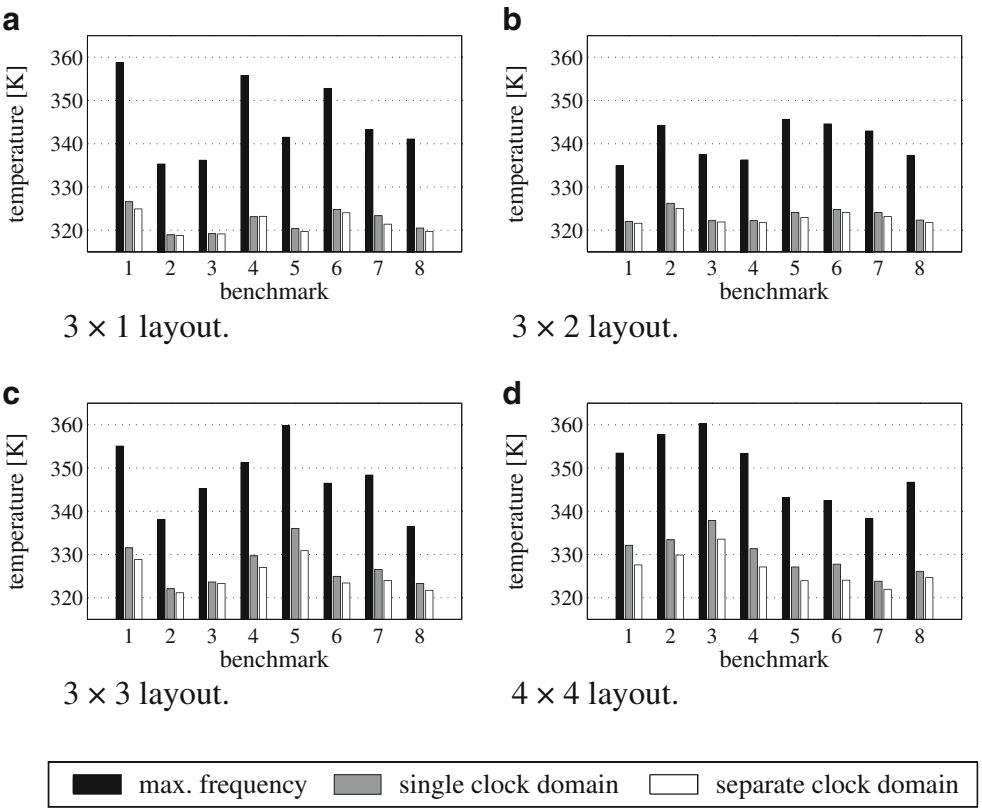

Fig. 8 Worst-case chip temperature for three different frequency configurations and four hardware platforms. The worst-case chip temperature is once calculated under the assumption that all processing components are running at maximum frequency, once under the assumption that the platform has a single clock domain, and once under the assumption that each processing component has a separate clock domain. **a** 3 × 1 layout. **b** 3 × 2 layout. **c** 3 × 3 layout. **d** 4 × 4 layout

ing components. Simulated annealing is used to solve the optimization problem in all benchmarks.

In Fig. 8, we plot the worst-case chip temperature for the three different frequency configurations and four hardware platforms. It shows that the worst-case chip temperature can be drastically reduced when the processing components are running at their optimal frequency. If each processing component has its own clock domain, the peak temperature is on average reduced by 24.2 K for the 3 × 1 layout, by 17.6 K for the 3 × 2 layout, 22.5 K for the 3 × 3 layout, and 22.8 K for the 4 × 4 layout.

7 Conclusion

In this paper, we presented a fast thermal analysis method to calculate an upper bound on the maximum temperature of a real-time application with non-deterministic workload running on a multi-core system. The considered thermal model is able to address various thermal effects like temperature-dependent leakage power and heat exchange between neighboring cores to accurately model the thermal behavior of multi-core systems. Afterwards, we applied the proposed thermal analysis method to calculate an optimal task assignment that minimizes the worst-case chip temperature and guarantees that all real-time deadlines are met. Finally, we have shown that the worst-case chip temperature can drastically be reduced when each core is running at its optimal operation frequency, i.e., the minimum operation frequency so that no real-time deadlines are missed.

Acknowledgments This work was supported by EU FP7 projects EURETILE and PRO3D, under grant numbers 247846 and 249776, and by the TRANSCEND Strategic Action from Nano-Tera.ch. Lars Schor was also partially supported by an Intel PhD Fellowship.

References

1. Bartolini A, Cacciari M, Tilli A, Benini L, Gries M (2010) A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores. In: Proc. Great Lakes symposium on VLSI (GLSVLSI), pp 311–316
2. Baruah S, Mok A, Rosier L (1990) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proc. real-time systems symposium (RTSS), pp 182–190
3. Benini L, Bertozzi D, Bogliolo A, Menichelli F, Olivieri M (2005) MPARM: exploring the multi-processor SoC design space with SystemC. J VLSI Signal Process 41(2):169–182
4. Bircher WL, John LK (2008) Analysis of dynamic power management on multi-core processors. In: Proc. int'l conf. on supercomputing (ICS), pp 327–338
5. Chakraborty S, Liu Y, Stoimenov N, Thiele L, Wandeler E (2006) Interface-based rate analysis of embedded systems. In: Proc. real-time systems symposium (RTSS), pp 25–34

6. Chantem T, Dick RP, Hu XS (2008) Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In: Proc. design, automation and test in Europe (DATE), pp 288–293
7. Coskun A, Rosing T, Whisnant K, Gross K (2008) Static and dynamic temperature-aware scheduling for multiprocessor SoCs. IEEE Trans Very Large Scale Integr (VLSI) Syst 16(9):1127–1140
8. Cui J, Maskell D (2012) A fast high-level event-driven thermal estimator for dynamic thermal aware scheduling. IEEE Trans Comput-Aided Des Integr Circ Syst 31(6):904–917
9. Donald J, Martonosi M (2006) Techniques for multicore thermal management: classification and new exploration. In: Proc. int'l symposium on computer architecture (ISCA), pp 78–88
10. Fisher N, Chen JJ, Wang S, Thiele L (2009) Thermal-aware global real-time scheduling on multicore systems. In: Proc. real-time and embedded technology and applications symposium (RTAS), pp 131–140
11. Henia R, Hamann A, Jersak M, Racu R, Richter K, Ernst R (2005) System level performance analysis—the SymTA/S approach. IEEE Proc Comput Digit Tech 152(2):148–166
12. Huang W, Ghosh S, Velusamy S, Sankaranarayanan K, Skadron K, Stan M (2006) HotSpot: a compact thermal modeling methodology for early-stage VLSI design. IEEE Trans Very Large Scale Integr (VLSI) Syst 14(5):501–513
13. Isci C, Buyuktosunoglu A, Cher CY, Bose P, Martonosi M (2006) An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget. In: Proc. int'l symposium on microarchitecture (MICRO), pp 347–358
14. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
15. Künzli S, Hamann A, Ernst R, Thiele L (2007) Combined approach to system level performance analysis of embedded systems. In: Proc. int'l conf. on hardware/software codesign and system synthesis (CODES+ISSS), pp 63–68
16. Liu Y et al (2007) Accurate temperature-dependent integrated circuit leakage power estimation is easy. In: Proc. design, automation and test in Europe (DATE), pp 1526–1531
17. Murali S, Mutapcic A, Atienza D, Gupta R, Boyd S, De Micheli G (2007) Temperature-aware processor frequency assignment for MPSoCs using convex optimization. In: Proc. int'l conf. on hardware/software codesign and system synthesis (CODES+ISSS), pp 111–116
18. Rabaey JM, Chandrakasan A, Nikolic B (2008) Digital integrated circuits, 3rd edn. Prentice Hall Press, Upper Saddle River
19. Schor L, Bacivarov I, Yang H, Thiele L (2012a) Fast worst-case peak temperature evaluation for real-time applications on multi-core systems. In: Proc. IEEE Latin American test workshop (LATW), pp 1–6
20. Schor L, Bacivarov I, Yang H, Thiele L (2012b) Worst-case temperature guarantees for real-time applications on multi-core systems. In: Proc. IEEE real-time and embedded technology and applications symposium (RTAS), pp 87–96
21. Skadron K, Stan MR, Sankaranarayanan K, Huang W, Velusamy S, Tarjan D (2004) Temperature-aware microarchitecture: modeling and implementation. ACM Trans Archit Code Optim 1(1):94–125
22. Sridhar M, Raj A, Vincenzi A, Ruggiero M, Brunschwiler T, Atienza Alonso D (2010) 3D-ICE: fast compact transient thermal modeling for 3D-ICs with inter-tier liquid cooling. In: Proc. int'l conf. on computer-aided design (ICCAD), pp 463–470
23. Thiele L, Chakraborty S, Naedele M (2000) Real-time calculus for scheduling hard real-time systems. In: Proc. IEEE int'l symposium on circuits and systems (ISCAS), pp 101–104
24. Thiele L, Schor L, Yang H, Bacivarov I (2011) Thermal-aware system analysis and software synthesis for embedded multiprocessors. In: Proc. design automation conference (DAC), pp 268–273
25. Thiele L, Schor L, Bacivarov I, Yang H (2013) Predictability for timing and temperature in multiprocessor system-on-chip platforms. ACM Trans Embed Comput Syst (TECS) 12(S1):48:1–48:25
26. Garcia del Valle P, Atienza D (2010) Emulation-based transient thermal modeling of 2D/3D systems-on-chip with active cooling. Microelectron J 41(10):1–9
27. Wandeler E, Thiele L (2006) Real-Time Calculus (RTC) toolbox. http://www.mpa.ethz.ch/Rtctoolbox
28. Wandeler E, Maxiaguine A, Thiele L (2006a) Performance analysis of greedy shapers in real-time systems. In: Proc. design, automation and test in Europe (DATE), pp 444–449
29. Wandeler E, Thiele L, Verhoef M, Lieverse P (2006b) System architecture evaluation using modular performance analysis: a case study. Int J Softw Tools Technol Transf 8(6):649–667
30. Xie Y, WL Hung (2006) Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design. J VLSI Signal Process 45(3):177–189
31. Yang CY, Chen JJ, Thiele L, Kuo TW (2010) Energy-efficient real-time task scheduling with temperature-dependent leakage. In: Proc. design, automation and test in Europe (DATE), pp 9–14

Lars Schor is a Ph.D. student at the Computer Engineering and Networks Laboratory of ETH Zurich, Switzerland. His research interests include multi-processor systems and thermal analysis methods for embedded real-time systems. He received a B.Sc. and M.Sc degree in computer engineering from ETH Zurich in 2011. In the same year, he received the "Willi Studer Price" and the "ETH Medal", both from ETH Zurich. In 2012, he received the "Intel Doctoral Student Honor Award".

Iuliana Bacivarov received the electrical engineering degree in 2002 from the National Polytechnic Institute of Bucharest, Romania. In 2002–2003, she received a master's degree in microelectronics integrated systems design from the Université Joseph Fourier in Grenoble, France, as well as a master's degree in quality and reliability engineering from the National Polytechnic Institute of Bucharest. She received her Ph.D. degree in microelectronics from the National Polytechnic Institute of Grenoble in 2006. She has been a post-doctoral researcher at the Computer Engineering and Networks Laboratory of ETH Zurich since 2006. Her research interests include design, analysis, and optimization of MPSoC.

Hoeseok Yang received the B.S. and Ph.D. degrees in computer science and engineering from the Seoul National University, Seoul, Korea, in 2003 and 2010, respectively. He is currently a post-doctoral researcher at the Computer Engineering and Networks Laboratory of ETH Zurich, Switzerland. His research interests include design, analysis, and optimization of MPSoC.

Lothar Thiele joined ETH Zurich, Switzerland, as a full professor of computer engineering in 1994, where he currently leads the Computer Engineering and Networks Laboratory. He received his Diplom-Ingenieur and Dr.-Ing. degrees in Electrical Engineering from the Technical University of Munich in 1981 and 1985 respectively. His research interests include models, methods and software tools for the design of embedded systems, embedded software and bioinspired optimization techniques. In 1986 he received the "Dissertation Award" of the Technical University of Munich, in 1987, the "Outstanding Young Author Award" of the IEEE Circuits and Systems Society, in 1988, the Browder J. Thompson Memorial Award of the IEEE, and in 2000–2001, the "IBM Faculty Partnership Award". In 2004, he joined the German Academy of Sciences Leopoldina. In 2005, he was the recipient of the Honorary Blaise Pascal Chair of University Leiden, The Netherlands. Since 2009 he is a member of the Foundation Board of Hasler Foundation, Switzerland. Since 2010, he is a member of the Academia Europaea.