Thermal-Aware Task Assignment for Real-Time Applications on Multi-Core Systems

Lars Schor, Hoeseok Yang, Iuliana Bacivarov, and Lothar Thiele

Computer Engineering and Networks Laboratory,
ETH Zurich, 8092 Zurich, Switzerland
firstname.lastname@tik.ee.ethz.ch

Abstract. The reduced feature size of electronic systems and the demand for high performance lead to increased power densities and high chip temperatures, which in turn reduce the system reliability. Thermal-aware task allocation and scheduling algorithms are promising approaches to reduce the peak temperature of multi-core systems with real-time constraints. However, as long as the worst-case chip temperature is not incorporated into system analysis, no guarantees on the performance can be given. This paper explores thermal-aware task assignment strategies for real-time applications with non-deterministic workload that are running on a multi-core system. In particular, tasks are assigned to the multi-core system so that the worst-case chip temperature is minimized and all real-time deadlines are met. Each core has its own clock domain and the static assigned frequency corresponds to the minimum operation frequency such that no real-time deadline is missed. Finally, we show that the proposed temperature minimization problem can efficiently be solved by metaheuristics.

Keywords: Real-Time Systems, Worst-Case Chip Temperature, Task Assignment, Thermal Analysis, Multi-Core Systems

1 Introduction

Multi-core systems outperform single-core platforms by offering higher performance and better power efficiency. However, the demand for increased performance and the reduced feature sizes lead to increasing power densities and high chip temperatures, which in turn reduce the system reliability. For example, exceeding the chip's peak temperature could lead to a reduction of performance or even damage the physical system. Reactive thermal management mechanisms, cooling systems, and thermal-aware task allocation and scheduling algorithms are potential techniques to tackle thermal and reliability issues.

Cooling systems for embedded real-time systems have to be designed for the worst-case chip temperature, i.e., the maximum chip temperature under all feasible scenarios of task arrivals. As the packaging costs of cooling systems increase super-linearly in power consumption [1], its design might be very expensive without the use of other thermal management mechanisms. Reactive thermal management mechanisms such as DVFS [2, 3] are widely used to address thermal issues. Despite their thermal effectiveness, these techniques cause a significant degradation of performance or lead to an expensive run-time overhead, both unacceptable in today's embedded real-time systems.

Various thermal-aware task allocation and scheduling algorithms have recently been studied [4–7]. However, as long as the worst-case chip temperature is not incorporated into system analysis, no guarantees on performance can be given and violations of real-time deadlines cannot be ruled out. Consequently, this paper explores thermal-aware task assignment and frequency selection strategies to reduce the worst-case chip temperature under real-time constraints. In particular, we consider the following problem:

> *Given are a set of tasks that are mapped onto a multi-core chip. Then, the goal is to assign each processing component its optimal frequency and to select a static assignment of tasks to processing components such that all real-time deadlines are met and the worst-case chip temperature is minimized.*

To this end, we propose a thermal analysis method to calculate a non-trivial upper bound on the maximum temperature of an embedded real-time system with multiple cores and non-deterministic workload that is later incorporated into the task assignment problem. Arrival curves from real-time calculus [8] are used to upper bound the task's workload in any time interval. Each processing component executes at a static frequency assigned at compile-time. This frequency is selected such that the real-time deadlines of all tasks are met and the worst-case chip temperature is minimized. The considered thermal model is able to address various thermal effects like the heat exchange between neighboring cores and temperature-dependent leakage power. The contributions of this paper can be summarized as follows:

- A novel method to calculate the worst-case chip temperature of an embedded real-time system with multiple cores and non-deterministic workload is formally derived.
- The minimization of the worst-case chip temperature with respect to real-time constraints is formulated as a nonlinear binary integer problem.
- We show the viability of the proposed methods in various case studies on hardware platforms with up to 16 cores.

The remainder of the paper is organized as follows: First, the considered problem is motivated by an introductory example in Section 2. Afterwards, in Section 3, the thermal and computational models considered in this paper are introduced. In Section 4, we discuss a method to calculate the worst-case chip temperature and show how to select the optimal operation frequency. The optimal task assignment problem is formulated as a nonlinear binary integer problem in Section 5. Finally, Section 6 presents case studies to highlight the viability of our methods and related work is discussed in Section 7.

2 Motivational Example

System Description. In order to motivate the considered problem, we examine various task to processing component assignments of a simple system with two identical tasks ν_1 and ν_2, and three homogeneous processing components with a maximum operation frequency of 1.6 GHz. The chip floorplan corresponds to the one outlined in Fig. 2. The parameters of the thermal model and the power dissipation parameters are summarized in Table 1(a) and Table 1(b). Both tasks have an invocation interval of 200 ms, a jitter of 400 ms, and a computational demand of $5 \cdot 10^7$ cycles, i.e., 31.25 ms when the processing component is running at its maximum operation frequency. Furthermore, the real-time deadline of a task is equal to its period.

Maximum Operation Frequency. First, we suppose that each processing component can only process at its maximum operation frequency, i.e., 1.6 GHz. Then, we calculate the worst-case chip temperature, i.e., the maximum chip temperature under all feasible scenarios of task arrivals for different mappings by the method proposed in Section 4.1. Mapping both tasks ν_1 and ν_2 to the same processing component results in a worst-case chip temperature of 339.22 K while mapping both tasks to different adjoined and non-adjoined processing components leads to a worst-case chip temperature of 343.61 K and 340.47 K, respectively. Note that the worst-case chip temperature is higher when both tasks are assigned to different processing components as both processing components are concurrently processing in the thermal critical scenario.

Optimal Operation Frequency. Next, the operation frequency of every processing component is the minimum frequency such that all deadlines are just met, calculated by (21). When both tasks are mapped onto the same processing component, the frequency can be reduced to 1.49 GHz and the maximum temperature of the system is 335.69 K. Assigning both tasks to different adjoined and non-adjoined processing components results in a worst-case chip temperature of 322.21 K and 321.73 K, respectively. As the operation frequency can be reduced to 0.74 GHz when both tasks are mapped onto different processing components, we observe the lowest worst-case chip temperature when both tasks are mapped onto non-adjoined processing components. In particular, the worst-case chip temperature was reduced by almost 22 K by selecting an adequate task to processing component assignment and optimal operation frequencies.

3 System Model and Problem Definition

In this section, the task, power, and temperature models are described.

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

3.1 Task Model

The task model considered in this paper is based on real-time calculus [8]. Let $\boldsymbol{\nu}$ be the set of tasks that are executed. We suppose that task ν_j is a stream of events and has a total workload of $R_{\nu_j}(s,t)$ cycles in time interval $[s,t)$. Each event has to complete its execution within D_{ν_j} time units after its arrival. The arrival curve α_{ν_j} upper bounds all possible cumulative workloads:

$$R_{\nu_j}(s,t) \leq \alpha_{\nu_j}(t-s) \quad \forall 0 \leq s < t \tag{1}$$

with $\alpha_{\nu_j}(\Delta) = 0$ for all $\Delta \leq 0$. In other words, $R_{\nu_j}(0,t)$ is the cumulative number of computing cycles of all events arrived in $[0,t)$. Arrival curves are a generalization of various well-know event arrival models as, for example, periodic event arrivals with jitter [9].

We use the concept of a demand bound function [10] to model the maximum resource demand of a task, and later to check the schedulability. In particular, the demand bound function $\mathrm{dbf}_{\nu_j}(\Delta)$ of task ν_j is:

$$\mathrm{dbf}_{\nu_j}(\Delta) = \alpha_{\nu_j}(\Delta - D_{\nu_j}) \quad \forall \Delta \geq 0 \ . \tag{2}$$

In other words, the maximum accumulated computational demand of all events that arrive and have deadline in any interval of length Δ does not exceed $\mathrm{dbf}_{\nu_j}(\Delta)$.

3.2 Processor Model

We consider a homogeneous multi-core system with a set of processing components $\boldsymbol{\Theta}$. The total accumulated workload of Θ_ℓ at time t is denoted as $R_\ell(0,t)$ and is, in any time interval of length $\Delta \leq t$, upper bounded by the arrival curve α_ℓ [8]:

$$R_\ell(t-\Delta,t) \leq \alpha_\ell(\Delta) = \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j,\Theta_\ell) \cdot \alpha_{\nu_j}(\Delta) \tag{3}$$

with the assignment function:

$$\Gamma(\nu_j,\Theta_\ell) = \begin{cases} 1 & \text{if } \nu_j \text{ executes on processing component } \Theta_\ell \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Likewise, the demand bound function $\mathrm{dbf}_\ell(\Delta)$ of a processing component Θ_ℓ can be calculated. For example, suppose that an earliest-deadline-first (EDF) scheduler runs on each processing component to arbitrate between events of different tasks assigned to the same processing component. Then, the demand bound function $\mathrm{dbf}_\ell(\Delta)$ is [10]:

$$\mathrm{dbf}_\ell(\Delta) = \sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j,\Theta_\ell) \cdot \mathrm{dbf}_{\nu_j}(\Delta) \ . \tag{5}$$

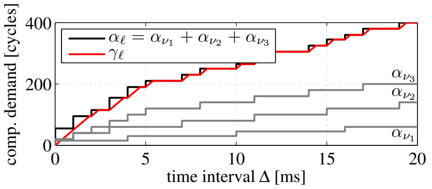


Fig. 1. Typical arrival curve $\alpha_\ell(\varDelta)$ with its corresponding upper bound on the accumulated computing time $\gamma_\ell(\varDelta)$.

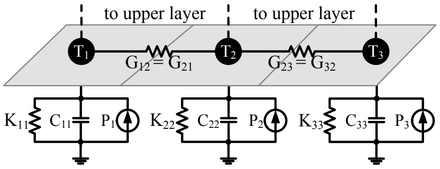


Fig. 2. *RC* circuit of the silicon layer for a chip with three processing components.

Each processing component executes at a static frequency f_ℓ with $0 < f_\ell \leq f_\ell^{\max}$. Therefore, the cumulated number of available computing cycles in time interval $[s, t)$ is $W_\ell(s,t) = f_\ell \cdot (t-s)$. If there are no waiting or arriving tasks in $[s,t)$, the available resources $W_\ell(s,t)$ are wasted. Otherwise, they are used to process incoming and waiting events.

The accumulated computing time $Q_\ell(0,t)$ describes the amount of cycles that processing component $\varTheta_\ell$ is spending to process an incoming workload of $R_\ell(0,t)$ time units. Using arrival curve $\alpha_\ell(\varDelta)$, the accumulated computing time $Q_\ell(t-\varDelta, t)$ can be upper bounded by $\gamma_\ell(\varDelta)$ for all intervals of length $\varDelta \leq t$ [11]:

$$Q_\ell(t-\varDelta, t) \leq \gamma_\ell(\varDelta) = \inf_{0 \leq \lambda \leq \varDelta} \left\{ \alpha_\ell(\lambda) + W_\ell(0, \varDelta - \lambda) \right\} \quad . \tag{6}$$

From the properties of the arrival curve, it follows that $\gamma_\ell(\varDelta)$ is monotonically increasing. The operation mode of a component can be expressed by the mode function $S_\ell(t)$, which is $S_\ell(t) = 1$ if the component is in 'active' processing mode at time t and $S_\ell(t) = 0$ if the component is in 'idle' processing mode at time t:

$$S_\ell(t) = \frac{\mathrm{d}Q_\ell(0,t)}{\mathrm{d}t} \cdot \frac{1}{f_\ell} = \begin{cases} 1 & \varTheta_\ell \text{ is processing some events at time } t \\ 0 & \varTheta_\ell \text{ is 'idle' at time } t. \end{cases} \tag{7}$$

A typical arrival curve and its corresponding upper bound on the accumulated computing time are outlined in Fig. 1.

The results of the paper also hold for heterogeneous platforms, but the task model becomes more complex. The workload $R_{\nu_j}(s,t)$ would just specify the number of events in time interval $[s,t)$. To calculate the accumulated computing time $Q_\ell(s,t)$, the workload of task ν_j is multiplied by the computation-time in cycles of an event of task ν_j when ν_j is assigned to processing component $\varTheta_\ell$.

3.3 Power Model

The power consumption of a processing component is the sum of the dynamic and leakage power consumption [4, 12]. Whenever a component is processing some events, the component is in 'active' mode, and consumes both dynamic and leakage power. Otherwise, it is in 'idle' mode, and consumes only leakage power.

Each processing component Θ_ℓ has its own clock domain and we suppose that the dynamic power consumption $P_{\ell,\text{dyn}}$ of component Θ_ℓ growths quadratically with its supply voltage v_ℓ and linearly with its operation frequency f_ℓ [13]:

$$P_{\ell,\text{dyn}}(t) \propto v_\ell^2 \cdot f_\ell \cdot S_\ell(t) \tag{8}$$

where the mode function $S_\ell(t)$ implies that $P_{\ell,\text{dyn}}(t) = 0$ if the component is in 'idle' processing mode. Note that the results of the paper also hold for other relations between supply voltage and frequency as long as they are monotone.

The leakage power consumption $P_{\ell,\text{leak}}$ of component Θ_ℓ is super linearly dependent on the temperature that can approximately be modeled by a linear function of the temperature [6, 14]:

$$P_{\ell,\text{leak}}(t) = \phi_{\ell\ell} \cdot T_\ell(t) + \psi_\ell \tag{9}$$

with T_ℓ the temperature of processing component ℓ, and the constants $\phi_{\ell\ell}$ and ψ_ℓ. We assume that the square of the supply voltage scales linearly with the operation frequency [5] and therefore, the total power consumption is:

$$\mathbf{P}(t) = \mathbf{P}_\text{dyn}(t) + \mathbf{P}_\text{leak}(t) = \boldsymbol{\phi} \cdot \mathbf{T}(t) + \boldsymbol{\rho} \cdot \text{diag}(\mathbf{f})^3 \cdot \mathbf{S}(t) + \boldsymbol{\psi} \tag{10}$$

with the diagonal matrix diag($\mathbf{f}$) of vector $\mathbf{f}$ and a constant diagonal matrix $\boldsymbol{\rho}$.

3.4 Temperature Model

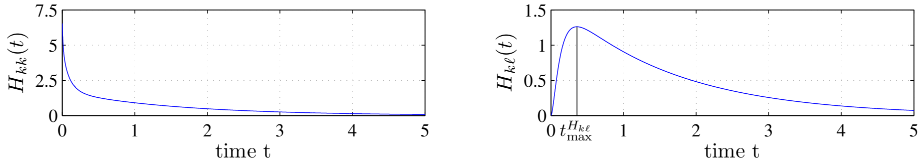
We model the temperature evolution of a multi-core system by an equivalent RC circuit [4, 15–17]. The vertical layout of the chip is modeled by four layers, namely the heat sink, heat spreader, thermal interface, and silicon die. Each layer is divided into a set of blocks according to architecture-level units, i.e., processing components, see Fig. 2 for the RC circuit of the silicon layer for a chip with three processing components. Every block is then mapped onto a node of the thermal circuit. The number of nodes and therefore, the order of the thermal model is $n = 4 \cdot |\Theta|$. In particular, the n-dimensional temperature vector $\mathbf{T}(t)$ at time t is described by a set of first-order differential equations:

$$\mathbf{C} \cdot \frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \left(\mathbf{P}(t) + \mathbf{K} \cdot \mathbf{T}^\text{amb}\right) - (\mathbf{G} + \mathbf{K}) \cdot \mathbf{T}(t) \tag{11}$$

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

Rewriting (11) with (10) leads to the state-space representation of the thermal model:

$$\frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \mathbf{A} \cdot \mathbf{T}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{12}$$



(a) Self impulse response $H_{kk}(t)$.

(b) General impulse response $H_{k\ell}(t)$.

Fig. 3. Impulse responses.

with input vector $\mathbf{u}(t) = \boldsymbol{\rho} \cdot \mathrm{diag}(\mathbf{f})^3 \cdot \mathbf{S}(t) + \boldsymbol{\psi} + \mathbf{K} \cdot \mathbf{T}^{\mathrm{amb}}$, $\mathbf{A} = -\mathbf{C}^{-1} \cdot (\mathbf{G} + \mathbf{K} - \boldsymbol{\phi})$, and $\mathbf{B} = \mathbf{C}^{-1}$. As the thermal system is linear and time-invariant, the temperature of node k is:

$$T_k(t) = T_k^{\mathrm{init}}(t) + \sum_{\ell=1}^{n} T_{k,\ell}(t) \tag{13}$$

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:

$$T_{k,\ell}(t) = \int_0^t H_{k\ell}(\xi) \cdot u_\ell(t-\xi)\,\mathrm{d}\xi \tag{14}$$

with

$$u_\ell(t) = \rho_{\ell\ell} \cdot f_\ell^3 \cdot S_\ell(t) + \psi_\ell + K_{\ell\ell} \cdot T^{\mathrm{amb}} = \rho_{\ell\ell} \cdot f_\ell^3 \cdot S_\ell(t) + u_\ell^{\mathrm{idle}} \;\; . \tag{15}$$

Nodes that do not correspond to a processing component have input $u_\ell = u_\ell^{\mathrm{idle}} = \psi_\ell + K_{\ell\ell} \cdot T^{\mathrm{amb}}$. Similar to [17], we assume that $H_{k\ell}(t)$ is a non-negative unimodal function that has its maximum at time $t_{\max}^{H_{k\ell}}$, see Fig. 3 for an illustration.

4 System Analysis

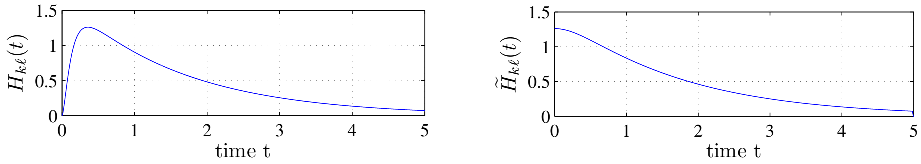
In this section, we propose a novel method to calculate the worst-case chip temperature of a multi-core system with non-deterministic workload and show how to select the operation frequencies in an optimal manner.

4.1 Peak Temperature Analysis

Suppose that the thermal RC network of a multi-core system is composed of n nodes. Then, the worst-case chip temperature T_S^* of a multi-core system is the maximum temperature of all individual nodes:

$$T_S^* = \max\left(T_1^*, \ldots, T_n^*\right) \tag{16}$$

where T_k^* is the worst-case peak temperature of node k. The following theorem follows from the results of [17] and states that the worst-case peak temperature is composed of $n + 1$ summands, that can be calculated individually.



(a) Original impulse response $H_{k\ell}(t)$.

(b) Non-increasing left-continuous sorting $\widetilde{H}_{k\ell}(t)$.

Fig. 4. Example of a typical impulse response $H_{k\ell}(t)$ and its non-increasing left-continuous sorting equivalent $\widetilde{H}_{k\ell}(t)$.

Theorem 1. *Suppose that* $T^*_{k,\ell}(\tau) = \max_{u_\ell \in U_\ell} (T_{k,\ell}(\tau))$ *with* U_ℓ *the set of all possible inputs* u_ℓ*,* $T_{k,\ell}(t)$ *defined as in (14), and a certain time instance* τ*. Then, an upper bound on the maximum temperature of node* k *at time* τ *is:*

$$T^*_k(\tau) \leq T^{\text{init}}_k + \sum_{\ell=1}^{n} T^*_{k,\ell}(\tau) \tag{17}$$

where n *is equal the number of nodes of the thermal RC circuit.*

Proof. Rewriting (13) with $T^*_{k,\ell}(\tau) = \max_{u_\ell \in U_\ell} (T_{k,\ell}(\tau))$ leads to:

$$\begin{aligned} T^*_k(\tau) &= \max_{\mathbf{u} \in \mathbf{U}} (T_k(\tau)) = \max_{\mathbf{u} \in \mathbf{U}} \left(T^{\text{init}}_k + \sum_{\ell=1}^{n} T_{k,\ell}(\tau) \right) \\ &\leq T^{\text{init}}_k + \sum_{\ell=1}^{n} \max_{u_\ell \in U_\ell} (T_{k,\ell}(\tau)) = T^{\text{init}}_k + \sum_{\ell=1}^{n} T^*_{k,\ell}(\tau). \quad \square \end{aligned} \tag{18}$$

As $T^*_{k,\ell}$ only depends on the workload of Θ_ℓ, we can individually maximize $T_{k,\ell}(\tau)$ at time instance τ for each processing component Θ_ℓ. The remaining question is how to calculate an upper bound $T^*_{k,\ell}(\tau)$ on $T_{k,\ell}(\tau)$ for a given time instance τ and all possible input sequences u_ℓ. To this end, we first introduce the non-increasing left-continuous sorting $\widetilde{H}_{k\ell}(t)$ of $H_{k\ell}(t)$ [18]. Roughly speaking, $\widetilde{H}_{k\ell}(t)$ is $H_{k\ell}(t)$ sorted in non-increasing order, see Fig. 4 for an example of a typical impulse response $H_{k\ell}(t)$ and its sorted equivalent $\widetilde{H}_{k\ell}(t)$. For illustration, we suppose discrete time, i.e., $H_{k\ell}(t)$ may change values only at multiples of δ and is constant for $t \in [r \cdot \delta, (r+1) \cdot \delta)$ for all $r \geq 0$. Then, $\widetilde{H}_{k\ell}[r]$ has the same elements as $H_{k\ell}[r]$, however, they are ordered non-increasingly, i.e., $\widetilde{H}_{k\ell}[r] \geq \widetilde{H}_{k\ell}[r+1]$ for all $r \geq 0$.

The next theorem shows that $T^*_{k,\ell}(\tau)$ is obtained by first calculating the sorted equivalent $\widetilde{H}_{k\ell}(t)$ of $H_{k\ell}(t)$, and then by convoluting $\widetilde{H}_{k\ell}(t)$ with $S^*_\ell(t)$. $S^*_\ell(t)$ is the mode function defined as in (7) resulting from the accumulated computing time $Q^*_\ell(0,t) = \gamma_\ell(\tau) - \gamma_\ell(\tau - t)$ for all $0 \leq t \leq \tau$. $Q^*_\ell(0,t)$ is called the thermal critical accumulated computing time. In particular, $Q^*_\ell(0,t)$ shifts

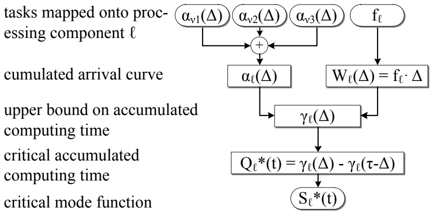

tasks mapped onto processing component ℓ
$\alpha_{v1}(\Delta)$
$\alpha_{v2}(\Delta)$
$\alpha_{v3}(\Delta)$
f_ℓ
+
cumulated arrival curve
$\alpha_\ell(\Delta)$
$W_\ell(\Delta) = f_\ell \cdot \Delta$
upper bound on accumulated computing time
$\gamma_\ell(\Delta)$
critical accumulated computing time
$Q_\ell*(t) = \gamma_\ell(\Delta) - \gamma_\ell(\tau-\Delta)$
critical mode function
$S_\ell*(t)$

Fig. 5. Steps to calculate the critical accumulated computing time $Q^*_\ell(0, t)$ and the critical mode function $S^*_\ell(t)$.

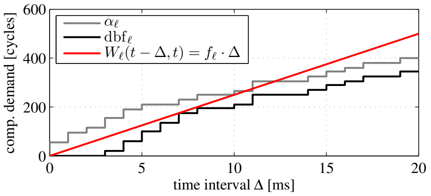


Fig. 6. Example of calculating the minimum operation frequency of component ℓ. $\alpha_\ell(\Delta)$ is the arrival curve defined as in Fig. 1, $\mathrm{dbf}_\ell(\Delta)$ the demand bound function, and $W_\ell(t - \Delta, t) = f_\ell \cdot \Delta$ the resulting available computing resources.

the computing time as late as possible to observation time τ. The steps required to calculate $Q^*_\ell(0, t)$ are detailed in Section 3 and summarized in Fig. 5.

Theorem 2. *Suppose that $\widetilde{H}_{k\ell}(t)$ is the non-increasing left-continuous sorting of $H_{k\ell}(t)$ [18], $Q^*_\ell(0, t) = \gamma_\ell(\tau) - \gamma_\ell(\tau - t)$ for all $0 \le t \le \tau$, and $T_{k,\ell}(t)$ is defined as in (14). Then, for any given time instance τ, $T^*_{k,\ell}(\tau)$ defined as:*

$$T^*_{k,\ell}(\tau) = u^{\mathrm{idle}}_\ell \cdot \int_0^\tau H_{k\ell}(t - \xi)\,\mathrm{d}\xi + \rho_{\ell\ell} \cdot f^3_\ell \cdot \int_0^\tau S^*_\ell(\xi) \cdot \widetilde{H}_{k\ell}(\tau - \xi)\,\mathrm{d}\xi \tag{19}$$

with $S^*_\ell(t) = \frac{\mathrm{d}Q^*_\ell(0,t)}{\mathrm{d}t}$ is an upper bound on $T_{k,\ell}(\tau)$, i.e., $T^*_{k,\ell}(\tau) \ge T_{k,\ell}(\tau)$.

Proof. The proof of this theorem is in the Appendix. □

So far, we have shown how to calculate an upper bound on the maximum temperature $T^*_k(\tau)$ of processing component k at time τ. However, we did not dwell on the amount of the observation time τ. The following theorem states that increasing the observation time τ will not decrease the worst-case peak temperature if $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$, where $(\mathbf{T}^\infty)^i$ is the steady-state temperature vector if all components are in 'idle' processing mode.

Theorem 3. *Suppose that $T^*_k(\tau)$ defined as in (17) is an upper bound on the maximum temperature of processing component k at time τ. Then, $T^*_k(\tau) \ge T_k(t)$ for all $0 \le t \le \tau$ and for any set of feasible workload traces with the same initial temperature vector $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$.*

Proof. With $T^*_{k,\ell}(\tau)$ defined as in (19), the proof is equivalent to the proof of Lemma 6 in [17]. □

In summary, Theorems 1 to 3 form together a method to calculate a non-trivial upper bound on the maximum temperature of a multi-core chip. First, we individually calculate $T^*_{k,\ell}(\tau)$ for all k, ℓ by (19). Then, in a second step, the maximum temperature $T_k^*(\tau)$ of each node k is calculated by (17). Finally, the worst-case chip temperature T_S^* follows from (16). The proposed method provides a safe bound on the maximum temperature, i.e., the method guarantees that the actual chip temperature will never exceed the temperature T_S^*.

4.2 Optimal Frequency Assignment

As a frequency reduction always leads to an accumulated computing time that is in all time intervals $\Delta \geq 0$ smaller or equal the original accumulated computing time, a frequency reduction results in a lower worst-case chip temperature. Therefore, the optimal frequency of every processing component Θ_ℓ is the minimum operation frequency such that no real-time deadline is missed. In particular, Θ_ℓ is schedulable, i.e., the real-time deadlines of all events are met, if the cumulated number of available computing resources W_ℓ is in no time interval Δ smaller than the maximum resources demand dbf_ℓ defined as in (5) [10]:

$$\mathrm{dbf}_\ell(\Delta) \leq W_\ell(t-\Delta, t) = f_\ell \cdot \Delta \quad \forall \Delta \geq 0 \ . \tag{20}$$

Therefore, the minimum operation frequency f_ℓ of processing component Θ_ℓ, such that all real-time deadlines are met, is:

$$f_\ell = \sup_{\Delta \geq 0} \left\{ \frac{\mathrm{dbf}_\ell(\Delta)}{\Delta} \right\} \ . \tag{21}$$

In other words, the frequency is selected such that the computing resource curve $W_\ell(t-\Delta, t) = f_\ell \cdot \Delta$ upper bounds the maximum resources demand $\mathrm{dbf}_\ell(\Delta)$ in every time interval $\Delta \geq 0$. From a geometric point of view, the problem is equivalent to determine a tangent to the curve $\mathrm{dbf}_\ell(\Delta)$ that crosses the origin. Practically, the optimal frequency can be calculated by the RTC toolbox [19]. Figure 6 illustrates this calculation with the help of an example.

5 Optimal Task Assignment

In this section, the optimal task assignment formulation is stated that solves the problem defined in Section 1.

5.1 Temperature Minimization Problem

In the last section, we proposed a method to calculate the optimal operation frequency of each processing component. Now, we apply these results to calculate an optimal task assignment that minimizes the worst-case chip temperature and guarantees that all real-time deadlines are met. If the worst-case chip temperature is smaller than the critical chip temperature, the system can safely execute the assignment without involving other (dynamic) thermal management strategies, that may lead to unpredictable behavior.

The objective of the temperature minimization problem (TMP) is to reduce the worst-case chip temperature:

$$\text{minimize } T_S^* = \max\left(T_1^*, \ldots, T_n^*\right) \tag{22}$$

where $T_k^* \geq T_k(t)$ for all $t \geq 0$ is the worst-case peak temperature of node k and n the number of nodes of the thermal RC circuit of the chip.

Furthermore, the operation frequency f_ℓ of processing component Θ_ℓ has to fulfill the following constraints following from (2) , (5) and (21):

$$f_\ell = \sup_{\Delta \geq 0} \left\{ \frac{\sum_{j=1}^{|\boldsymbol{\nu}|} \Gamma(\nu_j, \Theta_\ell) \cdot \alpha_{\nu_j}(\Delta - D_{\nu_j})}{\Delta} \right\} \leq f_\ell^{\max} \quad . \tag{23}$$

This constraint is also used to guarantee the schedulability. Whenever the operation frequency f_ℓ is smaller or equal to the maximum frequency $f_\ell^{\max}$ of processing component ℓ, the considered task assignment is schedulable, and otherwise, the task assignment is infeasible.

Finally, we have to guarantee that all tasks are assigned to exactly one processing component:

$$\sum_{\ell=1}^{|\boldsymbol{\Theta}|} \Gamma(\nu_j, \Theta_\ell) = 1 \quad \forall \nu_j \in \boldsymbol{\nu} \quad . \tag{24}$$

5.2 Evaluating a Task Assignment

So far, we formulated the TMP as a nonlinear binary optimization problem. Next, we will describe how to apply the methods presented in Section 4 to verify the schedulability and, if the task assignment is feasible, to calculate an upper bound on the maximum chip temperature of a task assignment.

The proposed method is summarized in Fig. 7. First, the demand bound function related to every processing component is individually computed by (5). Then, the minimum operation frequency f_ℓ is calculated for all processing components Θ_ℓ by (21) and the schedulability is tested. In particular, the system is only schedulable if all frequencies f_ℓ are smaller or equal their maximum frequency $f_\ell^{\max}$. Once we know that the system is schedulable with a particular set of operation frequencies, we calculate the worst-case chip temperature. To this end, the critical mode function S_ℓ^* is calculated for all processing components Θ_ℓ by the approach shown in Fig. 5. Finally, the worst-case chip temperature T_S^* follows by (16).

5.3 Efficient Temperature Reevaluation

Calculating the optimal task assignment might be expensive when all steps described in Section 5.2 are repeated for every possible assignment. Next, we show how to efficiently calculate the worst-case chip temperature of multiple related task assignments as it is the case in many metaheuristics. In a first step, we

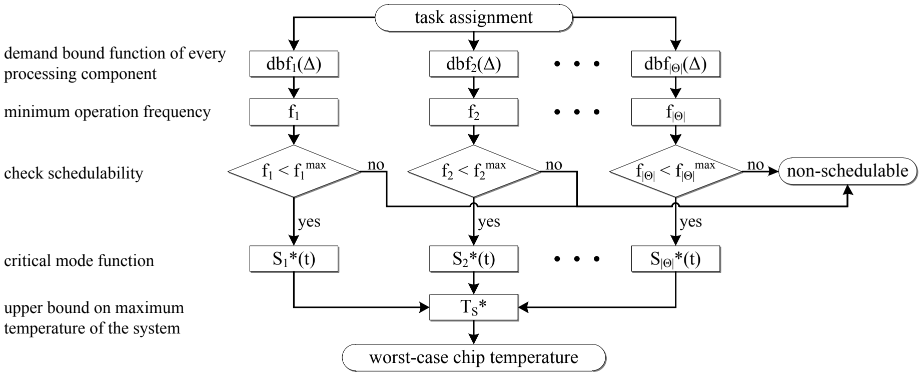


Fig. 7. Overview of the flow to analyze a single task assignment for schedulability and worst-case chip temperature.

rewrite the formula to calculate an upper bound on the maximum temperature of a node k as two terms, thereof one is assignment-independent, i.e., reusable, and one is assignment-dependent. Rewriting (17) with (19) leads to:

$$T_k^*(\tau) \leq T_k^{\text{init}} + \sum_{\ell=1}^{n} \left(u_\ell^{\text{idle}} \cdot \int_0^\tau H_{k\ell}(t-\xi)\,\mathrm{d}\xi \right) + \sum_{\ell=1}^{n} \left(\rho_{\ell\ell} \cdot f_\ell^3 \cdot \int_0^\tau S_\ell^*(\xi) \cdot \widetilde{H}_{k\ell}(\tau-\xi)\,\mathrm{d}\xi \right) = T_k^{\text{const}} + \sum_{\ell=1}^{|\Theta|} M_{k,\ell}\left(S_\ell^*\right)$$

with the number of nodes n of the thermal RC circuit, $T_k^{\text{const}} = T_k^{\text{init}} + \sum_{\ell=1}^{n} \left(u_\ell^{\text{idle}} \cdot \int_0^\tau H_{k\ell}(t-\xi)\,\mathrm{d}\xi \right)$ and $M_{k,\ell}\left(S_\ell^*\right) = \rho_{\ell\ell} \cdot f_\ell^3 \cdot \int_0^\tau S_\ell^*(\xi) \cdot \widetilde{H}_{k\ell}(\tau-\xi)\,\mathrm{d}\xi$. In the last step, we used the fact, that $M_{k,\ell}$ is all zero if node ℓ does not correspond to a processing component, to change the bound of summation. T_k^{const} is independent of the assignment, thus it is calculated once for all possible task assignments. Suppose that the worst-case chip temperature is calculated for two task assignments that only differ in the assignment of task ν_y. In particular, the first assignment maps task ν_y to component i and the second assignment maps task ν_y to component j. After calculating the worst-case chip temperature for the first assignment, the only elements that have to be recalculated for the second task assignment are all $M_{*,i}$ and $M_{*,j}$. In particular, the number of elements to be recalculated is reduced by a factor of $\frac{|\Theta|-2}{|\Theta|}$.

6 Case Studies

In order to study the viability of the proposed approaches, we solved TMP with four different solvers for different task sets and floorplans. To this end, the discussed methods are implemented in the real-time calculus toolbox [19].

Table 1. Thermal configuration of HotSpot and the power dissipation parameters of the power model defined as in (10).

(a) Thermal configuration of HotSpot.

Parameter	Symbol	Value
Silicon thermal cond. [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

(b) Power configuration.

Parameter	Value
$\phi_{\ell\ell}$ [W/K]	0.0228
$\rho_{\ell\ell}$ [W/GHz^3]	3.936
ψ_{ℓ} [W]	-2.756

6.1 System Description

We are targeting a reconfigurable homogeneous multi-core ARM platform with a variable number of processing components. The maximum frequency of all cores is 1.6 GHz and an EDF scheduler is running on each core to arbitrate between events of different tasks assigned to the same core. HotSpot [16] is used to calculate the thermal parameters of the platform, i.e., $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{K}$ matrices, see Table 1(a) for the detailed thermal configuration. The temperature-dependency of leakage power is addressed by linearizing the model described in [15] and the parameters of the power model for the platform with a 3×1 core layout are summarized in Table 1(b). As we consider a homogeneous platform, every component has the same power values. In all experiments, the traces start from the steady-state temperature in 'idle' mode, i.e., $\mathbf{T}^0 = (\mathbf{T}^\infty)^i$.

6.2 Performance of Four Different TMP Solvers

The optimal solution of the TMP can exhaustively be calculated for small task sets and platforms with a low number of processing components. We assess the performance of metaheurstics compared to exhaustive search for three different platforms and six different sets of tasks. The considered hardware platforms have a 3×1, 2×2, and 3×2 layout with three, four, and six cores, respectively. The event model of task ν_j is described using a set of two parameters, namely the period p_{ν_j} and jitter j_{ν_j} [9]. In particular, the period p_{ν_j} is uniformly chosen from $[1, 400\,]ms$, its jitter j_{ν_j} is uniformly chosen from $[1\,\mathrm{ms}, 2 \cdot p_{\nu_j}]$, and the computational demand is uniformly chosen from $\left[1, p_{\nu_j} \cdot f^{\max}/5\right]$ cycles with $f^{\max} = 1.6\,\mathrm{GHz}$. The real-time deadline of a task is equal to its period. Finally, the number of tasks in one set is randomly chosen between four and six tasks.

In total, we evaluate the performance of four different solvers for TMP. The first one exhaustively tests all possible assignments to compute the optimal solution from the TMP formulation. The optimal solution is compared, on the

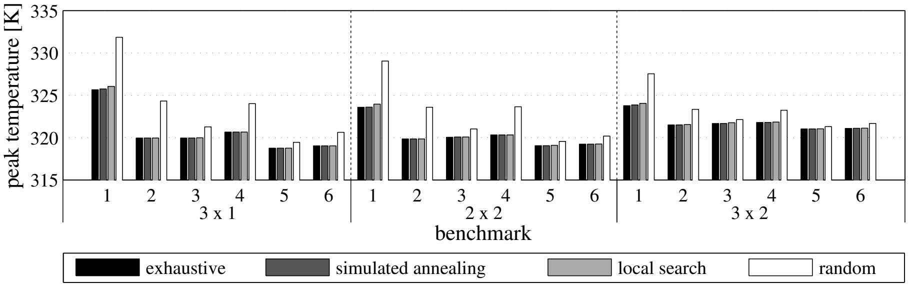
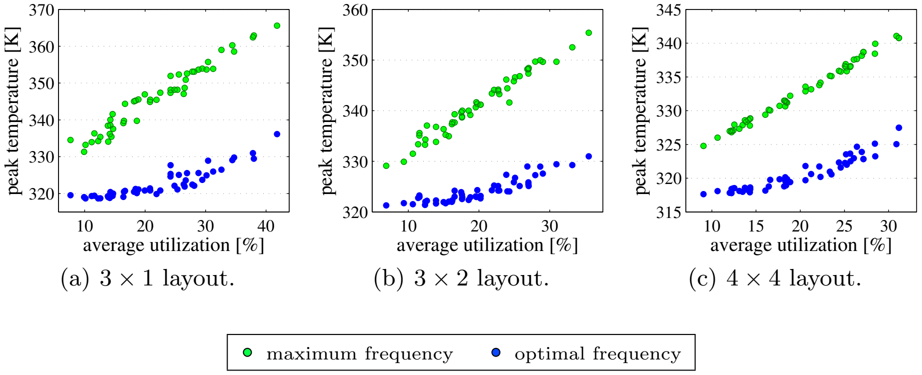


Fig. 8. Performance of four different TMP solvers. The hardware platforms have a 3×1, 2×2, and 3×2 layout, respectively.

one hand, to the solution of simulated annealing [20] and, on the other hand, to the solution of a local search algorithm, and the average peak temperature of 20 feasible random assignments. The local search algorithm fully exploits the characteristics of the considered peak temperature computation algorithm as described in Section 5.3 by testing the peak temperature of all neighbor assignments, and then, selecting the assignment that minimizes the peak temperature the most.

Fig. 8 compares the performance of the four different solvers. The peak temperature of the optimal task assignment is on average 2.27 K lower than the peak temperature of the random assignments. Simulated annealing found in all considered benchmarks an assignment that has a peak temperature that is no more than 0.11 K higher than the optimal assignment. This shows that probabilistic metaheuristics are well suited to solve TMP. The local search algorithm calculates an assignment that is no more than 0.4 K higher than the peak temperature of the optimal assignment. In particular, one can see that there are a few task sets where the local search algorithm proposes a task assignment that is significantly worse than the optimal solution. This could be prevented by extending the local search algorithm such that it does not only consider the direct neighborhood of the current assignment, but all assignments up to its k-th neighborhood. The difference in terms of peak temperature between the solvers becomes even larger if the number of tasks per task set is increased as more local optima emerge. As frequency reduction damps the effect of burst on the peak temperature, most solvers are able to find an acceptable solution. However, once the damping is removed, the peak temperature might drastically increase, which in turn results in higher peak temperature differences between the solvers.

On a 2.55 GHz Intel Core i5-2400S processor, calculating the optimal solution for the hardware platform with 6 cores took on average 1.52 h. Simulated annealing and the local search algorithm finished on average in 22.6 s and 0.77 s, respectively. Finally, calculating the peak temperature of 20 random assignments took on average 3.1 s.



(a) 3 × 1 layout.

(b) 3 × 2 layout.

(c) 4 × 4 layout.

Fig. 9. Worst-case chip temperature for three hardware platforms. To calculate the worst-case peak temperature, TMP is once solved under the assumption that all processing components are running at maximum frequency, and once under the assumption that the components are running at optimal frequency.

6.3 Performance for Different Utilizations and Floorplans

In the second case study, we evaluate the worst-case chip temperature for different floorplans and utilizations. The layout of the considered platforms is 3 × 1, 3 × 2, and 4 × 4 with 3, 6, and 16 cores, respectively. In all benchmarks, the TMP is solved by simulated annealing. The task sets are iteratively generated, starting with an initial size of $|\Theta|$ randomly generated tasks. Then, as long as the system is schedulable, we add a new randomly generated task to the collection. In total, we generate 50 different task sets for each hardware platform.

For each benchmark, we resolve the TMP once under the assumption that all processing components are running at maximum frequency, i.e., 1.6 GHz, and once under the assumption that the components are running at their optimal frequency such that each benchmark is characterized by a triple $(T^*_{f_{\max}}, T^*_{f_{\mathrm{opt}}}, util)$. $T^*_{f_{\max}}$ is the peak temperature when the components are running at maximum frequency, $T^*_{f_{\mathrm{opt}}}$ is the peak temperature when the components are running at optimal frequency and *util* is the average utilization of all cores when the components are running at their optimal frequency and the jitter is ignored. Even thought the components are running at their optimal frequency, the utilization is not 100 % as the jitter has a high impact on the selection of the frequencies.

Finally, in Fig. 9, we plot $T^*_{f_{\max}}$ and $T^*_{f_{\mathrm{opt}}}$ as a function of *util* for three hardware platforms. It shows that the chip temperature can drastically be reduced when the processing components are running at their optimal frequency. In particular, the peak temperature can be reduced on average by 23.6 K for the 3 × 1 layout, by 17.0 K for the 3 × 2 layout, and by 12.1 K for the 4 × 4 layout. Furthermore, Fig. 9 shows that the worst-case chip temperature does not necessarily increase with the utilization as different amounts of non-determinism might cause higher chip temperatures for lower utilizations.

7 Related Work

Xie and Hung [21] were the first to identify the topic of thermal-aware task allocation and scheduling. Later, a convex optimization technique for temperature-aware frequency assignment is proposed to maximize the performance under temperature constraints [5] and the task scheduling problem is statically solved using integer linear programming for minimizing energy, and reducing hot spots [7].

The minimization of the peak temperature in the presence of real-time deadlines is formulated as a nonlinear programming problem in [22]. A mixed-integer linear programming formulation for assigning and scheduling tasks with hard real-time constraints to reduce the peak temperature is proposed in [4]. Finally, Fisher et al. [6] proposed a global scheduling algorithm such that all cores are running at their ideally preferred speed, and the peak temperature is minimized. However, as the peak temperature is calculated in these works by either steady-state temperature analysis or transient temperature evolution, the proposed methods cannot be used to optimize the task to processing component assignment of a system with non-deterministic workload and hard real-time guarantees. As high chip temperatures can significantly reduce the system's performance, real-time constraints can only be guaranteed if the worst-case chip temperature is incorporated in real-time analysis, at design-time.

HotSpot [16] is the most popular simulator for thermal analysis. However, as thermal simulation methods only cover a fraction of all possible system behaviors, they are not able to capture the maximum temperature of an application with non-deterministic workload. Tackling this challenge, a method to calculate the worst-case chip temperature of a multi-core system with non-deterministic workload has been proposed in [17]. In comparison with the method proposed in this paper, the authors of [17] use periodic event streams with burst [9] for the event model of every processing component.

8 Conclusion

In this paper, we formulated the thermal-aware task assignment and frequency selection problem to optimize the worst-case chip temperature under real-time constraints as a nonlinear binary integer problem. In order to solve the proposed problem, we described a novel analytical method to calculate an upper bound on the maximum chip temperature under all feasible scenarios of task arrivals. Each core has its own clock domain and the static assigned frequencies correspond to the minimum operation frequencies such that no real-time deadline is missed. The considered thermal model is able to address various thermal effects like heat exchange between neighboring cores and temperature-dependent leakage power. Arrival curves from real-time calculus are used to upper bound the task's workload in any time interval. Case studies have shown that the worst-case chip temperature of an embedded multi-core system can be reduced by more than 20 K by assigning each processing component its ideally preferred frequency and selecting the optimal task to processing component assignment.

Acknowledgments. This work was supported by EU FP7 projects EURETILE and PRO3D, under grant numbers 247846 and 249776.

References

1. Gunther, S., Binns, F., Carmean, D., Hall, J.: Managing the Impact of Increasing Microprocessor Power Consumption. Intel Technology Journal **5**(1) (2001) 1–9
2. Donald, J., Martonosi, M.: Techniques for Multicore Thermal Management: Classification and New Exploration. In: Proc. Int'l Symposium on Computer Architecture (ISCA), Boston, MA, USA, IEEE (2006) 78 – 88
3. Isci, C., Buyuktosunoglu, A., Cher, C.Y., Bose, P., Martonosi, M.: 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), IEEE (2006) 347–358
4. Chantem, T., Dick, R., Hu, X.: Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In: Proc. Design, Automation and Test in Europe (DATE), Munich, Germany, ACM/IEEE (2008) 288–293
5. Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., De Micheli, G.: Temperature-Aware Processor Frequency Assignment for MPSoCs Using Convex Optimization. In: Proc. Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Salzburg, Austria, ACM (2007) 111–116
6. Fisher, N., Chen, J.J., Wang, S., Thiele, L.: Thermal-Aware Global Real-Time Scheduling on Multicore Systems. In: Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS), San Francisco, USA, IEEE (2009) 131–140
7. Coskun, A., Rosing, T., Whisnant, K., Gross, K.: Static and Dynamic Temperature-Aware Scheduling for Multiprocessor SoCs. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **16**(9) (2008) 1127–1140
8. Thiele, L., Chakraborty, S., Naedele, M.: Real-Time Calculus for Scheduling Hard Real-Time Systems. In: Proc. Int. Symposium on Circuits and Systems (ISCAS). Volume 4., Geneva, Switzerland, IEEE (2000) 101–104
9. Henia, R., Hamann, A., Jersak, M., Racu, R., Richter, K., Ernst, R.: System Level Performance Analysis - The SymTA/S Approach. IEEE Proc. Comp. and Digital Tech. **152**(2) (2005) 148–166
10. Baruah, S., Mok, A., Rosier, L.: Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor. In: Proc. Real-Time Systems Symposium (RTSS), Lake Buena Vista, FL, USA, IEEE (1990) 182–190
11. Wandeler, E., Maxiaguine, A., Thiele, L.: Performance Analysis of Greedy Shapers in Real-Time Systems. In: Proc. Design, Automation and Test in Europe (DATE), Munich, Germany (2006) 444–449
12. Chen, J.J., Wang, S., Thiele, L.: Proactive Speed Scheduling for Real-Time Tasks under Thermal Constraints. In: Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS), San Francisco, CA, USA, IEEE (2009) 141–150
13. Rabaey, J.M., Chandrakasan, A., Nikolic, B.: Digital Integrated Circuits. 3rd edn. Prentice Hall Press (2008)
14. Liu, Y., Dick, R.P., Shang, L., Yang, H.: Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. In: Proc. Design, Automation and Test in Europe (DATE), Nice, France (2007) 1526–1531
15. Skadron, K., et al.: Temperature-Aware Microarchitecture: Modeling and Implementation. ACM Trans. Architec. Code Optim. **1**(1) (2004) 94–125

16. Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., Stan, M.: HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **14**(5) (2006) 501–513
17. Schor, L., Bacivarov, I., Yang, H., Thiele, L.: Worst-Case Temperature Guarantees for Real-Time Applications on Multi-Core Systems. In: Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS), Beijing, China, IEEE (2012) 87–96
18. Ferreira, P.: Sorting Continuous-Time Signals: Analog Median and Median-Type Filters. IEEE Trans. Signal. Proces. **49**(11) (2001) 2734–2744
19. Wandeler, E., Thiele, L.: Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox` (2006)
20. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by Simulated Annealing. Science **220**(4598) (1983) 671–680
21. Xie, Y., Hung, W.l.: Temperature-Aware Task Allocation and Scheduling for Embedded Multiprocessor Systems-on-Chip (MPSoC) Design. The Journal of VLSI Signal Processing **45**(3) (2006) 177–189
22. Liu, Y., Yang, H., Dick, R., Wang, H., Shang, L.: Thermal vs Energy Optimization for DVFS-Enabled Processors in Embedded Systems. In: Proc. Int'l Symposium on Quality Electronic Design (ISQED), San Jose, CA, USA, IEEE (2007) 204–209

Appendix: Proof of Theorem 2

In the following, we will show that $T^*_{k,\ell}(\tau) \geq T_{k,\ell}(\tau)$ for any valid $T_{k,\ell}(\tau)$. Rewriting (14) with (15) leads to $T_{k,\ell}(t) = u_\ell^{\text{idle}} \cdot \int_0^t H_{k\ell}(t-\xi)\,\mathrm{d}\xi + \rho_{\ell\ell} \cdot f_\ell^3 \cdot \int_0^t S_\ell(\xi) \cdot H_{k\ell}(t-\xi)\,\mathrm{d}\xi$. Then we have:

$$T^*_{k,\ell}(\tau) - T_{k,\ell}(\tau) = \rho_{\ell\ell} \cdot f_\ell^3 \cdot \left(\int_0^\tau S^*_\ell(\xi) \cdot \widetilde{H}_{k\ell}(\tau-\xi)\,\mathrm{d}\xi - \int_0^\tau S_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\,\mathrm{d}\xi \right)$$

with $\rho_{\ell\ell} \cdot f_\ell^3 > 0$. In other words, we have to show that $\int_0^\tau S^*_\ell(\xi) \cdot \widetilde{H}_{k\ell}(\tau-\xi)\,\mathrm{d}\xi \geq \int_0^\tau S_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\,\mathrm{d}\xi$.

Discretization

In order to simplify the proof technicalities, we suppose discrete time, i.e., $S_\ell(t), S^*_\ell(t), \widetilde{H}_{k\ell}(t)$, and $H_{k\ell}(t)$ may change values only at multiples of δ and are constant for $t \in [r \cdot \delta, (r+1) \cdot \delta)$ for all $r \geq 0$. With $r_\tau = \tau \cdot \delta$, we have:

$$\int_0^\tau S^*_\ell(\xi) \cdot \widetilde{H}_{k\ell}(\tau-\xi)\,\mathrm{d}\xi = \delta \cdot \sum_{r=0}^{r_\tau - 1} S^*_\ell[r] \cdot \widetilde{H}_{k\ell}[r_\tau - 1 - r] \tag{25}$$

and

$$\int_0^\tau S_\ell(\xi) \cdot H_{k\ell}(\tau-\xi)\,\mathrm{d}\xi = \delta \cdot \sum_{r=0}^{r_\tau - 1} S_\ell[r] \cdot H_{k\ell}[r_\tau - 1 - r] \quad . \tag{26}$$

Next, we show that $\sum_{r=0}^{r_\tau-1} S_\ell\,[r] \cdot H_{k\ell}\,[r_\tau - 1 - r] \leq \sum_{r=0}^{r_\tau-1} S_\ell^*\,[r] \cdot \widetilde{H}_{k\ell}\,[r_\tau - 1 - r]$ for all S_ℓ that satisfy (6), by induction. To this end, we will prove that:

$$\underbrace{\sum_{r=w}^{w+\pi-1} S_\ell\,[r] \cdot H_{k\ell}\,[r_\tau - 1 - r]}_{\mathcal{T}(\pi,w,S_\ell)} \leq \underbrace{\sum_{r=r_\tau-\pi}^{r_\tau-1} S_\ell^*\,[r] \cdot \widetilde{H}_{k\ell}\,[r_\tau - 1 - r]}_{\mathcal{T}^*(\pi)} \tag{27}$$

for any $\pi \in [0, r_\tau]$ and any $w \in [0, r_\tau - \pi]$.

Base Case

First, we show that the statement is true for $\pi = 1$. Rewriting (27) with $\pi = 1$ leads to $S_\ell\,[w] \cdot H_{k\ell}\,[r_\tau - 1 - w] \leq S_\ell^*\,[r_\tau - 1] \cdot \widetilde{H}_{k\ell}\,[r_\tau - 1 - (r_\tau - 1)] = S_\ell^*\,[r_\tau - 1] \cdot \widetilde{H}_{k\ell}\,[0]$. As $S_\ell^*\,[r_\tau - 1] = 1$ and $\widetilde{H}_{k\ell}\,[0] \geq H_{k\ell}\,[\eta]$ for all $\eta \geq 0$, the statement is true for $\pi = 1$.

Induction Hypothesis

Next, we show that the statement is true for π if it is true for $\pi - 1$. In other words, we assume as *induction hypothesis* that:

$$\mathcal{T}(\pi - 1, w, S_\ell) \leq \mathcal{T}^*(\pi - 1) \tag{28}$$

holds for all $w \in [0, r_\tau - \pi + 1]$.

Induction Step

Let us prove by contradiction that (27) is true for any π. Therefore, assume for contradiction that there exists a $\overline{w}$ such that:

$$\mathcal{T}(\pi, \overline{w}, S_\ell) > \mathcal{T}^*(\pi)\ \ . \tag{29}$$

Now, we differ between the following cases:

Case 1: $S_\ell\,[\overline{w} + \pi - 1] = 0$.
The contradiction follows from $\mathcal{T}(\pi, \overline{w}, S_\ell) = \mathcal{T}(\pi - 1, \overline{w}, S_\ell) + S_\ell\,[\overline{w} + \pi - 1] \cdot H_{k\ell}\,[r_\tau - 1 - (\overline{w} + \pi - 1)] = \mathcal{T}(\pi - 1, \overline{w}, S_\ell) + 0 \cdot H_{k\ell}\,[r_\tau - 1 - (\overline{w} + \pi - 1)] \leq \mathcal{T}^*(\pi - 1) \leq \mathcal{T}^*(\pi)$.

Case 2: $S_\ell\,[\overline{w} + \pi - 1] = 1,\ S_\ell^*\,[r_\tau - \pi] = 1$.
As $\mathcal{T}^*(\pi) = \mathcal{T}^*(\pi - 1) + \widetilde{H}_{k\ell}\,[r_\tau - 1 - (r_\tau - \pi)]$ and $\mathcal{T}(\pi, \overline{w}, S_\ell) = \mathcal{T}(\pi - 1, \overline{w}, S_\ell) + H_{k\ell}\,[r_\tau - 1 - (\overline{w} + \pi - 1)]$, it follows that $\widetilde{H}_{k\ell}\,[\pi - 1] < H_{k\ell}\,[r_\tau - (\overline{w} + \pi)]$.
First, we show that $\widetilde{H}_{k\ell}\,[\pi - 1] < H_{k\ell}\,[r_\tau - (\overline{w} + \pi)]$ implies that $H_{k\ell}\,[r_\tau - \overline{w}] \leq \widetilde{H}_{k\ell}\,[\pi - 1]$. As $H_{k\ell}$ is a non-negative unimodal function, the condition $H_{k\ell}\,[r_\tau - \overline{w}] > \widetilde{H}_{k\ell}\,[\pi - 1]$ requires that all $\pi + 1$

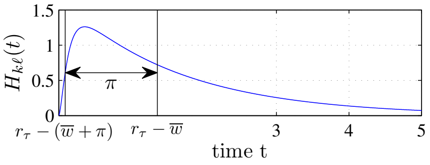


Fig. 10. Sketch of the proof that in Case 2, $\widetilde{H}_{k\ell}[\pi-1] < H_{k\ell}[r_\tau - (\overline{w}+\pi)]$ implies that $\widetilde{H}_{k\ell}[\pi-1] \geq H_{k\ell}[r_\tau - \overline{w}]$.

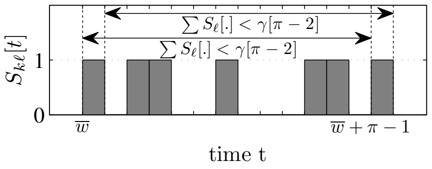


Fig. 11. Sketch of Case 3(b) that illustrates how the accumulated computing time is upper bounded.

elements $H_{k\ell}[\eta]$ for $\eta \in [r_\tau - (\overline{w}+\pi), r_\tau - \overline{w}]$ fulfill $H_{k\ell}[\eta] > \widetilde{H}_{k\ell}[\pi-1]$, see Fig. 10 for an illustration. However, as $\widetilde{H}_{k\ell}[\pi-1]$ is the π-th largest element of $H_{k\ell}$, this is a contradiction, and $H_{k\ell}[r_\tau - \overline{w}] \leq \widetilde{H}_{k\ell}[\pi-1]$. As $\mathcal{T}(\pi-1, w, S_\ell) \leq \mathcal{T}^*(\pi-1)$ for any w, in particular also for $w = \overline{w}+1$, we find $\mathcal{T}(\pi, \overline{w}, S_\ell) \leq H_{k\ell}[r_\tau - \overline{w}] + \mathcal{T}(\pi-1, \overline{w}+1, S_\ell) \leq \widetilde{H}_{k\ell}[\pi-1] + \mathcal{T}^*(\pi-1) = \mathcal{T}^*(\pi)$, which is a contradiction.

Case 3: $S_\ell[\overline{w}+\pi-1] = 1$, $S^*_\ell[r_\tau - \pi] = 0$.
From $S^*_\ell[r_\tau-\pi] = 0$ follows that $\sum_{r=r_\tau-\pi+1}^{r_\tau-1} S^*_\ell[r] = \gamma[\pi-2] = \sum_{r=r_\tau-\pi}^{r_\tau-1} S^*_\ell[r] = \gamma[\pi-1]$.

a) $S_\ell[\overline{w}] = 0$.
From $S_\ell[\overline{w}] = 0$ follows that $\mathcal{T}(\pi, \overline{w}, S_\ell) = \mathcal{T}(\pi-1, \overline{w}+1, S_\ell) \leq \mathcal{T}^*(\pi-1) = \mathcal{T}^*(\pi)$, which is a contradiction.

b) $S_\ell[\overline{w}] = 1$.
First note that $\sum_{r=\overline{w}}^{\overline{w}+\pi-1} S_\ell[r] \leq \gamma_\ell[\pi-1] = \gamma_\ell[\pi-2]$. As $S_\ell[\overline{w}] = 1$, we know that $\sum_{r=\overline{w}+1}^{\overline{w}+\pi-1} S_\ell[r] < \sum_{r=r_\tau-\pi+1}^{r_\tau-1} S^*_\ell[r] = \gamma_\ell[\pi-2]$ and as $S_\ell[\overline{w}+\pi-1] = 1$, we know that $\sum_{r=\overline{w}}^{\overline{w}+\pi-2} S_\ell[r] < \sum_{r=r_\tau-\pi+1}^{r_\tau-1} S^*_\ell[r] = \gamma_\ell[\pi-2]$, see also Fig. 11.
In case that $H_{k\ell}[r_\tau - 1 - \overline{w}] < H_{k\ell}[r_\tau - 1 - (\overline{w}+\pi-1)]$, we know that $H_{k\ell}[\eta] \geq H_{k\ell}[r_\tau - 1 - \overline{w}]$ for any $\eta \in [r_\tau - (\overline{w}+\pi), r_\tau - \overline{w} - 1]$ (see Fig. 10). Therefore, there exists:

$$\overline{S}_\ell[r] = \begin{cases} 0 & r = \overline{w} \\ 1 & r = \overline{w}' \\ S_\ell[r] & \text{otherwise} \end{cases} \tag{30}$$

with $\overline{w} < \overline{w}' < \overline{w}+\pi-1$ and $S_\ell[\overline{w}'] = 0$. As $H_{k\ell}[r_\tau - 1 - \overline{w}'] \geq H_{k\ell}[r_\tau - 1 - \overline{w}]$, we have $\mathcal{T}(\pi, \overline{w}, S_\ell) \leq \mathcal{T}(\pi, \overline{w}, \overline{S}_\ell)$. Similarly, we can find a $\overline{S}_\ell$ and $\overline{w}'$ for the case $H_{k\ell}[r_\tau - 1 - \overline{w}] \geq H_{k\ell}[r_\tau - 1 - (\overline{w}+\pi-1)]$.
Now, applying Case 1 or Case 3.a to $\overline{S}_\ell$ shows that $\mathcal{T}(\pi, \overline{w}, \overline{S}_\ell) \leq \mathcal{T}^*(\pi)$, and therefore, $\mathcal{T}(\pi, \overline{w}, S_\ell) \leq \mathcal{T}(\pi, \overline{w}, \overline{S}_\ell) \leq \mathcal{T}^*(\pi)$, which is the contradiction.

As we have shown that (27) is true for any π, it is particularly true for $\pi = r_\tau$, and the theorem follows. □