Predictability for Timing and Temperature in Multiprocessor System-on-Chip Platforms

LOTHAR THIELE, LARS SCHOR, IULIANA BACIVAROV,
and HOESEOK YANG, ETH Zurich

High computational performance in multiprocessor system-on-chips (MPSoCs) is constrained by the ever-increasing power densities in integrated circuits, so that nowadays MPSoCs face various thermal issues. For instance, high chip temperatures may lead to long-term reliability concerns and short-term functional errors. Therefore, the new challenge in designing embedded real-time MPSoCs is to guarantee the final performance and correct function of the system, considering both functional and non-functional properties. One way to achieve this is by ruling out mapping alternatives that do not fulfill requirements on performance or peak temperature already in early design stages. In this article, we propose a thermal-aware optimization framework for mapping real-time applications onto MPSoC platforms. The performance and temperature of mapping candidates are evaluated by formal temporal and thermal analysis models. To this end, analysis models are automatically generated during design space exploration, based on the same specifications as used for software synthesis. The analysis models are automatically calibrated with performance data reflecting the execution of the system on the target platform. The data is automatically obtained prior to design space exploration based on a set of benchmark mappings. Case studies show that the performance and temperature requirements are often conflicting goals and optimizing them together leads to major benefits in terms of a guaranteed and predictable high performance.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]: Realtime and embedded systems; C.4 [**Performance of systems**]: Measurement techniques

General Terms: Design, Performance, Reliability

Additional Key Words and Phrases: MPSoC, temperature analysis, design automation

ACM Reference Format:
Thiele, L., Schor, L., Bacivarov, I., and Yang, H. 2012. Predictability for timing and temperature in multiprocessor system-on-chip platforms. ACM Trans. Embedd. Comput. Syst. XX, XX, Article XX (2012), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Multiprocessor system-on-chips (MPSoCs) are a promising solution to keep pace with the ever-increasing demand of computational performance. But as the obtained performance imposes a major rise in power consumption per unit area, MPSoCs can have a high chip temperature problem. Nowadays, the thermal wall is recognized as one of the most significant barriers towards high performance systems [Hardavellas et al. 2011]. For example, high chip temperatures may lead to long-term reliability concerns and short-term functional errors. Therefore, providing guarantees on maximum temperature is as important as functional correctness and timeliness when designing em-

This research has been founded by EU FP7 projects EURETILE and PRO3D, under grant numbers 247846 and 249776.
Author's addresses: L. Thiele, L. Schor, I. Bacivarov, and H. Yang, Computer Engineering and Networks Laboratory, ETH Zurich, CH-8092 Zurich, Switzerland; email: firstname.lastname@tik.ee.ethz.ch.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

bedded real-time MPSoCs. Aware of performance-temperature dependency, we take the new challenge of optimizing the system design with respect to both performance and temperature. More specifically, we aim at ruling out mapping alternatives that do not conform to real-time and peak temperature requirements already in early design stages. Optimizing a system in both respects leads to major benefits in terms of a guaranteed and predictable high performance.

In this article, we present a high-level optimization framework for mapping real-time applications onto embedded MPSoC platforms that provides guarantees on both temporal and thermal correctness. The timing and temperatures of mapping candidates are evaluated by means of formal worst-case real-time analysis methods to provide safe bounds on the execution time and the maximum chip temperature. Practically, we extend the distributed operation layer (DOL) [Thiele et al. 2007; Huang et al. 2012] with the ability to perform not only performance analysis, that is done in modular performance analysis (MPA) [Wandeler et al. 2006], but also to relate it to high-level temperature analysis in design space exploration.

The method proposed in [Schor et al. 2012] to analyze the worst-case chip temperature of an MPSoC platform is implemented as an extension of MPA so that both temperature and real-time analysis are done within the same framework in order to speed up design space exploration. Furthermore, to explore the design space without user-interactions, the analysis models are automatically generated from the same set of specifications as used for software synthesis. To increase the model accuracy, the analysis models are calibrated with data corresponding to the target platform. The data is obtained in an automatic manner by either simulation on a virtual platform or execution on the real hardware, prior to design space exploration.

Based on a prototype implementation of the proposed high-level mapping optimization framework, we demonstrate that there is no single optimal solution with respect to both real-time system performance and temperature, but all generated solutions are worst-case guaranteed with respect to application behavior and impact of a non-deterministic environment. With the proposed framework, designers receive a powerful tool to map applications onto MPSoC platforms so that the system can safely execute without further involving other (dynamic) thermal management strategies, which may lead to unpredictable behavior. The contributions of this paper can be summarized as follows:

— A systematic approach to integrate worst-case real-time and peak temperature analysis methods into a unique framework for MPSoC platforms is developed.
— To guarantee the accuracy of system analysis, we show that the proposed timing and thermal analysis models can be generated from the same set of specifications as used for software synthesis without user-interaction and automatically calibrated with performance data corresponding to the target platform.
— The viability of the proposed approach is demonstrated by the integration into a high-level optimization framework for mapping real-time applications onto embedded MPSoC platforms, namely DOL. Finally, the framework is applied to realistic case studies to provide guarantees on temporal and thermal correctness of the system, and make timing and temperature aware decisions.

The remainder of the article is organized as follows: First, related work is discussed in the next section. Afterwards, Section 3 presents the mapping optimization framework including the fully automated design flow. Section 4 introduces the computational and thermal models used for formal system analysis. The thermal analysis method is described in Section 5. The automated generation and calibration of the thermal analysis model is presented in Section 6 and finally, Section 7 presents case studies to highlight the viability of our methods.

2. RELATED WORK

By providing automatic mapping of applications onto distributed platforms, model-based frameworks outperform heuristic-driven approaches in terms of efficiency and profit for the design of embedded multiprocessor systems [Zhao et al. 2005]. Examples of model-based frameworks are Artemis [Pimentel 2008], Koski [Kangas et al. 2006], or DOL [Thiele et al. 2007; Huang et al. 2012]. In order to predict timing behavior of embedded systems, all model-based frameworks include performance analysis of underlying multiprocessor systems. Often, analysis methods are classified based on their scope [Bacivarov et al. 2010]. For example, performance analysis can be based on simulation at different levels of abstractions or worst-case/best-case analysis. Simulation-based methods suffer various drawbacks over worst-case/best-case models as for example long run times and insufficient coverage of corner cases. Examples of best-case/worst-case analysis methods for multiprocessor systems are MPA [Wandeler et al. 2006] or SymTA/S [Henia et al. 2005] that provide upper and lower bounds on timing properties.

Thermal constraints are considered as the most significant barrier towards high performance in modern MPSoC platforms [Hardavellas et al. 2011]. Increased power densities induce hot spot heat fluxes leading to high chip-temperatures, which in turn cause long-term reliability concerns and short-term functional errors [Coskun et al. 2007]. Reduced on-chip wire length motivates the use of three-dimensional stacking in the future [Loh 2008], which even magnifies problems with temperature [Zhu et al. 2008]. Consequently, modern mapping optimization frameworks for real-time systems have to consider both timing metrics and the thermal characteristics of the system design. However, none of the above-mentioned model-based frameworks includes temperature analysis.

Nowadays, to address thermal issues, reactive thermal management techniques [Donald and Martonosi 2006; Brooks and Martonosi 2001; Kumar et al. 2006] are typically used. For example, multiple architectural-level techniques for thermal management like DVFS and stop-go scheduling are evaluated in [Donald and Martonosi 2006]. Causing a significant degradation of performance or leading to expensive run-time overhead, reactive thermal management techniques are often undesirable in today's embedded systems, in particular when tackling real-time constraints. On top of that, reactive thermal management techniques become more and more unfeasible as the operating voltage is limited by saturation [Watanabe et al. 2010]. The alternative is to adopt system-level mechanisms, and already include temperature analysis at design-time.

Thermal aware task allocation and scheduling algorithms for MPSoC platforms are explored in [Xie and Hung 2006; Coskun et al. 2007; Chantem et al. 2008]. In particular, thermal aware heuristics to reduce the maximum and average temperature are compared with power aware heuristics in [Xie and Hung 2006]. Thermal management techniques with unknown workload like load balancing or temperature aware random scheduling are discussed in [Coskun et al. 2007]. A mixed-integer linear programming formulation to reduce the peak temperature of an application is proposed in [Chantem et al. 2008]. All these design-time methods have in common that the temperature analysis is performed by either simulation or steady-state analysis. First, the transient power dissipation of the system is determined by means of a (power aware) simulator, either software-based [Bartolini et al. 2010; Thiele et al. 2011] or hardware-based [Garcia del Valle and Atienza 2010]. Afterwards, power dissipation is used to calculate either the average temperature based on a steady-state analysis [Yang et al. 2010] or to evaluate the transient temperature evolution in a thermal simulator. HotSpot [Huang et al. 2006; Skadron et al. 2004] and 3D-ICE [Sridhar et al.

2010] are the most common examples of such thermal simulators. 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 under all feasible scenarios of task arrivals. Consequently, simulation-based thermal analysis methods may lead to an undesired underestimation of the maximum temperature.

Instead of simulation, we use formal thermal analysis methods to predict the worst-case temperature. [Rai et al. 2011] proposed a method to calculate the worst-case peak temperature of a single-node system with non-deterministic workload. By incorporating the heat transfer among neighboring nodes, [Schor et al. 2012] proposed a method to calculate the worst-case chip temperature of an MPSoC platform. The article at hand extends this work by proposing a systematic approach to integrate worst-case chip temperature analysis into a model-based framework. Moreover, joint effects of performance and temperature have not been considered so far in any model-based framework for designing multiprocessor systems. By including timing and thermal analysis into the same framework, we provide to system designers a powerful decision tool, which includes both real-time and peak temperature analysis.

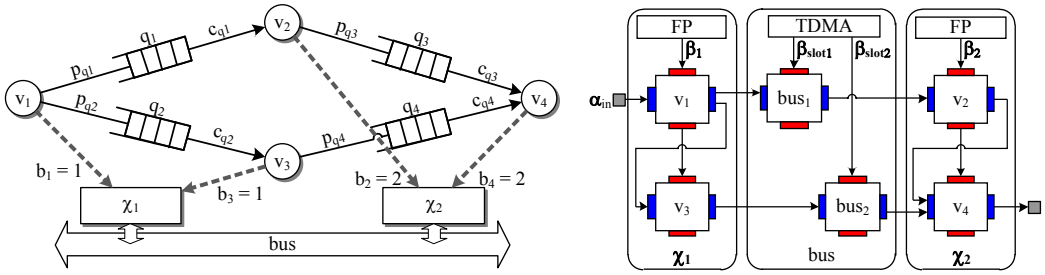
3. MAPPING OPTIMIZATION FRAMEWORK

To optimize the mapping of a streaming application onto an MPSoC platform with respect to both worst-case performance and worst-case chip temperature, we extend DOL [Thiele et al. 2007] that has already successfully been applied to many architectures like MPARM platform [Benini et al. 2005], Sony/Toshiba/IBM Cell BE [Kahle et al. 2005], or Atmel DIOPSIS 940 [Paolucci et al. 2006]. Following the Y-chart paradigm [Kienhuis et al. 1997], the application and architecture are specified separately, and then linked by a mapping. This separation of concerns enables the iterative improvement of the system by modifying either of these specifications. The rest of this section discusses first the application, architecture, and mapping models, and then, it introduces the design flow considered in this paper. In order to illustrate the used notation, an example system is illustrated in Fig. 1a.

3.1. Application Model

In the scope of this article, the synchronous data flow (SDF) [Lee and Messerschmitt 1987] model of computation is considered in order to specify the application behavior independently of the communication. More formally, we represent an SDF application as a directed, connected graph $\mathcal{A} = (V, Q, W)$ where every process $v \in V$ represents a node in the graph. Edges model unbounded channels $q \in Q$, which in turn are used by processes to communicate with each other. $W = \{w_1, \ldots, w_{|Q|}\}$ describes the token consumption behavior of channels. The token consumption w_q of channel q is represented as a triple (p_q, c_q, d_q). Assuming q connects nodes v_1 and v_2, the number of tokens produced by node v_1 in every execution is denoted p_q; c_q is the number of tokens consumed in every execution by v_2; and the initial amount of tokens in channel q is d_q.

In the SDF model of computation, tasks represent processes that execute concurrently. If they are mapped onto different processing elements, they can execute in parallel. Data dependencies are taken into account by means of edges that explicitly model the data flow between processes. A task is blocked as long as not all its predecessor tasks have produced the number of tokens the task needs to be triggered, that is, there is at least one incoming channels q, which contains less than c_q tokens. Once triggered, the task can process the received tokens independently of all other tasks and predecessor tasks can simultaneously start to process the next tokens. The separation of computation and communication has various advantages with respect to software development. In particular, for system analysis, this separation enables the modular



(a) Formal system specification. (b) MPA model.

Fig. 1. System specification and the corresponding MPA model.

generation and calibration of analysis models from the same specification as it is used for actual system synthesis.

As we will see in Section 4, the analysis model is based on MPA that abstracts the workload in any time interval $\Delta \geq 0$ by a so-called arrival curve. Therefore, the results of this paper hold for other model of computations, as well. In particular, the only requirement for the model of computation is that the workload of every independent component is bounded in any time interval $\Delta \geq 0$.

3.2. Architecture Model

Unlike temporal analysis, thermal analysis requires a detailed description of the architecture to model the heat flow between neighboring nodes. The architecture $\mathcal{T} = \{\chi_1, \ldots, \chi_n\}$ is assumed to be a heterogeneous multiprocessor system with n processing components connected by a shared bus or a network-on-chip. The placement of components is specified by a floorplan, see [Atienza et al. 2007] for examples of MPSoC floorplans.

3.3. Mapping Model

The mapping specification describes both the binding b of processes to architecture components and their scheduling σ on shared resources. The binding can be represented by a vector $b = (b_1; \ldots; b_{|V|}) \in \{1, \ldots, n\}^{|V|}$ where $b_i = k$ if process v_i is assigned to processing component χ_k. The scheduling policy is supposed to be work-conserving, that is, the processing component has to process as soon as there is data available. This assumption applies to most traditional scheduling algorithms as for example, earliest-deadline-first (EDF), rate-monotonic (RM), fixed-priority (FP), and deadline-monotonic (DM). For simplicity, we neglect the mapping of communication channels, and implicitly assume that they are mapped onto the same processing component as the sender process, e.g. local memory of the corresponding processor, even though our tool can deal with cases that are more general. The extension to mapping communication channels is obvious in general and only augments the dimensionality of the design space.

3.4. Design Flow

So far, we discussed the specification of the application, architecture, and mapping models, which form together the systems specification, that is, the input to software synthesis. While the application and architecture specifications are provided by the system designer, the mapping is automatically calculated during the design flow. Next, we describe the steps to explore the optimal mapping, that is, the binding and scheduling of a multi-processor streaming application onto an MPSoC platform in a time and thermal optimal manner. By varying the binding of application elements, that is, pro-

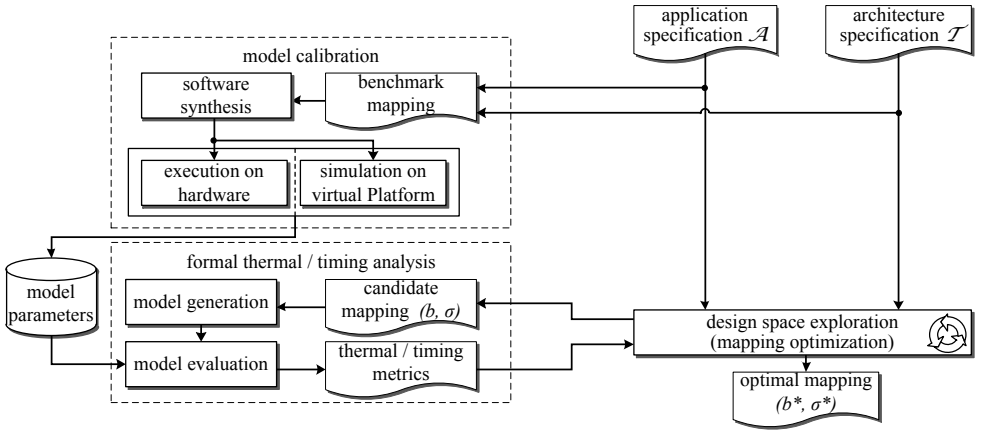


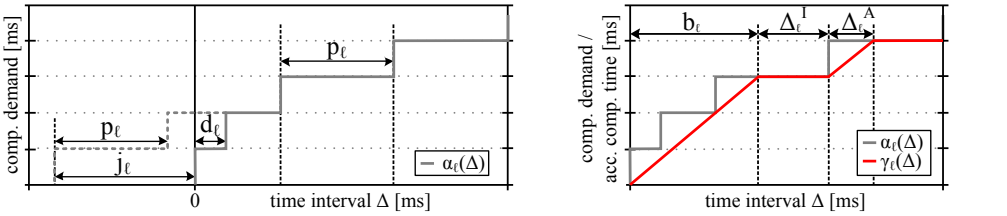
Fig. 2. Design flow for mapping optimization.

cesses $v \in V$ and channels $c \in C$, to computation and communication resources, aimed system properties are optimized. Figure 2 illustrates the design flow for mapping optimization as it is considered in this paper.

The design flow is composed of three major parts, namely model calibration, design space exploration, and thermal and timing analysis. During design space exploration, the thermal and timing characteristics of every candidate mapping (b, σ) are analyzed. The corresponding analysis models are parameterized with precalculated model parameters that are extracted during model calibration. The output of the considered design flow is the optimal mapping (b^*, σ^*) of the given multiprocessor application $\mathcal{A}$ onto the MPSoC architecture $\mathcal{T}$.

The acquisition of the required model parameters is denoted as model calibration and is detailed in Section 6.2. Model calibration is performed based on a set of benchmark mappings prior to analyzing the candidate mappings during design space exploration. First, a benchmark implementation is generated by synthesizing the benchmark system composed of the application, architecture, and benchmark mapping specification. Then, the benchmark implementation is simulated on a virtual platform or executed on the real hardware. The required parameters are automatically extracted and stored in a database for later use during thermal and timing analysis.

Once the model parameters are extracted, the design space exploration tool can start to explore for optimal mappings by analyzing the performance and chip temperature of various candidate mappings (b, σ). The proposed design flow bases the performance analysis on formal worst-case real-time analysis methods. In particular, modular performance analysis (MPA) [Wandeler et al. 2006; Huang et al. 2012] is used. In order to evaluate the performance of a single candidate mapping, the proposed mapping optimization framework first generates an abstract model out of the same set of specifications as used for software synthesis, namely application, architecture, and candidate mapping specification. Then, in a second step, the abstract model is examined with respect to timing and thermal characteristics. In particular, timing properties are analyzed by the methods described in [Huang et al. 2012] and the thermal properties by the methods proposed in Section 5. We argue in this article that the considered design flow can be completely automated. In other words, after specifying the input, that is, the application and the architecture, the design flow calculates a good mapping (b^*, σ^*) without user interaction.



(a) Arrival curve α_ℓ for PJD model. (b) Accumulated computing time function.

Fig. 3. Computational model.

The focus of this paper is on formal analysis methods, and their automated generation and calibration. Thus, we do not detail the part on how to calculate good candidate mappings, but refer to the summary given in Section 2.

4. SYSTEM MODEL FOR WORST-CASE TIMING AND TEMPERATURE ANALYSIS

This section introduces the formal models to analyze a streaming application on an MPSoC platform with respect to both timing and temperature.

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 denoted $H_{k\ell}$ and $\mathbf{T}$ denotes a vector whose k-th element is denoted T_k.

4.1. Computational Model

The computational model for timing and thermal analysis is based on MPA [Wandeler et al. 2006] that uses a compositional approach to split up a system into actors with small interference. After separately characterizing each actor, the system is analyzed with real-time calculus [Thiele et al. 2000], which is based on network calculus [Le Boudec and Thiran 2001]. Formally, we use a graph $\mathcal{M} = (\mathcal{V}, E)$ to represent an MPA model. The set of nodes $\mathcal{V}$ represents the actors and the set of edges E represents event streams (abstracted by arrival curves α) and resource streams (abstracted by service curves β). The data dependency between different actors is given by the execution sequence. Communication resources are described by the same concept as computational resources, namely by a cumulative function that defines the number of available resources in any time interval. Figure 1b shows the MPA model of the multiprocessor system from Fig. 1a. For illustration, we suppose fixed priority preemptive scheduling on all processors and TDMA scheduling on the bus even though other policies can be modeled, as well. α_{in} abstracts the system's input stream, β_1 and β_2 abstract the available resources of component χ_1 and χ_2. β_{slot_1} and β_{slot_2} abstract the resource availability of the bus. In the following, we will summarize the event stream and workload models relevant to calculate the worst-case chip temperature of a multiprocessor system.

The considered model of an event stream follows the ideas of real-time calculus. We suppose that in time interval $[s,t)$, events with a total workload of $R_\ell(s,t)$ time units arrive at component ℓ. Each event is supposed to have a constant workload of Δ_ℓ^A time units. 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 $\alpha_\ell(0) = 0$. For the scope of this article, we assume that an event stream can be specified by a period p_ℓ, a jitter j_ℓ, and a minimum interarrival distance d_ℓ [Henia et al. 2005], see Fig. 3a for a typical arrival curve.

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 queues. The resource availability of such a component ℓ is abstracted by the service curve $\beta_\ell(\Delta) = \Delta$ for all intervals of length $\Delta \geq 0$. The accumulated computing time $Q_\ell(s,t)$ describes the amount of time units that component ℓ is spending to process an incoming workload of $R_\ell(s,t)$ time units. Therefore, for work-conserving scheduling algorithms, the accumulated computing time $Q_\ell(s,t)$ in time interval $[s,t)$ is $Q_\ell(s,t) = \inf_{s \leq u \leq t} \{(t-u) + R_\ell(s,u)\}$ provided that there is no buffered workload in the ready queue at time s [Thiele et al. 2000]. Using arrival curve α_ℓ, the accumulated computing time $Q_\ell(t-\Delta,t)$ can be upper bounded by $\gamma_\ell(\Delta)$ for all intervals of length $\Delta < t$:

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

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. Whenever the slope is 1, the component is in 'active' processing mode, that is, it is processing events. When the slope is 0, the component is 'idle', that is, it is in sleep mode. The processing mode can be expressed by the mode function:

$$S_\ell(t) = \frac{\mathrm{d}Q_\ell(s,t)}{\mathrm{d}t} = \begin{cases} 1 & \text{component } \ell \text{ is 'active'}, \\ 0 & \text{component } \ell \text{ is 'idle'}. \end{cases} \tag{3}$$

A typical arrival curve and its corresponding upper bound on the accumulated computing time function are outlined in Fig. 3b. We characterize the upper bound on the computing time of component ℓ by the length b_ℓ of the first interval with slope 1, 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 bound on the accumulated computing time is selected so that all intervals with slope 0 have the same length.

4.2. Thermal Model

A well-accepted thermal model of an MPSoC architecture with $|\mathcal{T}|$ processing components is to describe the temperature evolution by means of an equivalent RC circuit [Krum 2000; Skadron et al. 2004; Huang et al. 2006; Chantem et al. 2008]. 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, that is, in our case, processing components. Figure 4 represents an example floorplan and its equivalent RC circuit of the silicon layer. 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 $n = 4 \cdot |\mathcal{T}|$. In particular, the n-dimensional temperature vector $\mathbf{T}(t)$ at time t is described by a set of first-order differential equations:

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

where $\mathbf{C}$ is 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 $t^0 = 0$. $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{K}$ are calculated from the floorplan and thermal configuration of the chip. $\mathbf{G}$ is a non-positive matrix whose diagonal elements are zero and $\mathbf{K}$ is a non-negative diagonal matrix.

(a) Floorplan of an MPSoC platform with two processing components.

(b) Equivalent RC circuit of one layer of the MPSoC platform in Fig. 4a.

Fig. 4. Floorplan and equivalent RC circuit.

We assume a linear dependency of power dissipation on temperature [Liu et al. 2007; Chantem et al. 2008] 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 with constant coefficients. As we suppose that the leakage power of a component is independent of its processing mode, we characterize the power dissipated by component ℓ in 'active' and 'idle' processing mode as:

$$P_\ell(t) = \begin{cases} P^a_\ell(t) = \phi_{\ell\ell} \cdot T_\ell + \psi^a_\ell & \text{if } S_\ell(t) = 1, \\ P^i_\ell(t) = \phi_{\ell\ell} \cdot T_\ell + \psi^i_\ell & \text{if } S_\ell(t) = 0. \end{cases} \tag{6}$$

Rewriting Eq. (4) with Eq. (5) leads to the state-space representation of the thermal model:

$$\frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} = \mathbf{A} \cdot \mathbf{T}(t) + \mathbf{B} \cdot \mathbf{u}(t) \tag{7}$$

where $\mathbf{u}(t) = \boldsymbol{\psi}(t) + \mathbf{K} \cdot \mathbf{T}^{\mathrm{amb}}$ is called the input vector, $\mathbf{A} = -\mathbf{C}^{-1} \cdot (\mathbf{G} + \mathbf{K} - \boldsymbol{\phi})$, and $\mathbf{B} = \mathbf{C}^{-1}$. As $\mathbf{A}$ and $\mathbf{B}$ are time-invariant, the considered thermal model represents a linear and time-invariant (LTI) system [Friedland 1986] and consequently for $t > 0$, a closed-form solution of the temperature yields:

$$\mathbf{T}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{T}^0 + \int_{-\infty}^{\infty} \mathbf{H}(t - \xi) \cdot \mathbf{u}(\xi)\, \mathrm{d}\xi \tag{8}$$

where $\mathbf{H}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{B}$. $H_{k\ell}(t)$ corresponds to the impulse response between node ℓ and node k. With $\mathbf{T}^{\mathrm{init}}(t) = e^{\mathbf{A} \cdot t} \cdot \mathbf{T}^0$, the temperature $T_k(t)$ of node k is of form:

$$T_k(t) = T^{\mathrm{init}}_k(t) + \sum_{\ell=1}^{n} T_{k,\ell}(t) \tag{9}$$

where $T_{k,\ell}(t)$ is the convolution between the impulse response $H_{k\ell}$ and the input u_ℓ:

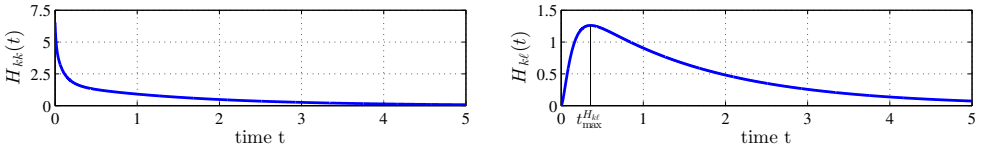
$$T_{k,\ell}(t) = \int_0^t H_{k\ell}(t - \xi) \cdot u_\ell(\xi)\, \mathrm{d}\xi. \tag{10}$$

By using the processing mode of component ℓ, we can associate the input u_ℓ of node ℓ with the workload of the corresponding component:

$$u_\ell(t) = S_\ell(t) \cdot u^a_\ell + (1 - S_\ell(t)) \cdot u^i_\ell. \tag{11}$$

where $u^a_\ell = \psi^a_\ell + K_{\ell\ell} \cdot T^{\mathrm{amb}}$ and $u^i_\ell = \psi^i_\ell + K_{\ell\ell} \cdot T^{\mathrm{amb}}$.

As the impulse response describes the reaction of the system over time, next, we will discuss its properties with respect to the considered thermal model. First, we note that



(a) Self-impulse response $H_{kk}(t)$. (b) General impulse response $H_{k\ell}(t)$.

Fig. 5. Impulse responses.

$\mathbf{H}(t) \geq \mathbf{0}$ for all $t \geq 0$, as $\mathbf{A}$ is essentially non-negative, that is, $\mathbf{A}_{k\ell} \geq 0$ for all $\ell \neq k$, which in turn leads to $e^{\mathbf{A}\cdot t} \geq \mathbf{0}$ [Birkhoff and Varga 1958]. The self-impulse response $H_{kk}(t)$ can be calculated by $H_{kk}(t) = e_k' \cdot e^{\mathbf{A}\cdot t} \cdot e_k \cdot B_{kk}$, where e_k is the unit vector pointing in k direction. Therefore, $H_{kk}(0) \geq H_{kk}(t) \geq 0$ for all $t \geq 0$ and $\frac{\mathrm{d}H_{kk}(t)}{\mathrm{d}t} \leq 0$ [Maeda et al. 1977], see Fig. 5a for an illustration. Next, based on various experiments and the self-impulse response, we conjecture that the general impulse response $H_{k\ell}(t)$ is a non-negative unimodal function that has its maximum at time $t_{\max}^{H_{kj}}$, that is, $\frac{\mathrm{d}H_{k\ell}(t)}{\mathrm{d}t} \geq 0$ for all $0 \leq t \leq t_{\max}^{H_{kj}}$ and $\frac{\mathrm{d}H_{k\ell}(t)}{\mathrm{d}t} \leq 0$ for all $t > t_{\max}^{H_{kj}}$ as illustrated in Fig. 5b. Intuitively, this can be motivated by the duality of a thermal network and a grounded capacitor RC circuit. We know from [Gupta et al. 1997] that all impulse responses of a stable RC tree network are unimodal. As a particular path of a general RC network usually dominates the impulse response, we can neglect local maximums that are caused by different paths without significantly affecting the resulting temperature. In all performed experiments, we never detected a departure from this conjecture. With temperature in mind, this describes that temperature rises with power on the component that produces power without a delay and on a neighbor of the component that produces power only after a delay.

5. TEMPERATURE ANALYSIS

The framework presented in this article uses formal thermal analysis to calculate the worst-case chip temperature, that is, the maximum temperature of a chip under all feasible scenarios of task arrivals. In this section, the thermal analysis method to calculate the worst-case chip temperature is described. We refer the reader to [Schor et al. 2012] for detailed proofs of the theorems stated in this section.

Clearly, the worst-case chip temperature T_S^* of an MPSoC platform is the maximum temperature of all individual nodes:

$$T_S^* = \max\left(T_1^*, \ldots, T_n^*\right) \tag{12}$$

where T_k^* is the worst-case peak temperature of node k and n the number of nodes. We call the set of cumulative workload traces $\mathbf{R}$ that leads to the worst-case peak temperature T_k^* of node k the *critical* set of cumulative workload traces and denote it with $\mathbf{R}^{\{k\}}$. Because temperature rises with power consumed at another node only after a delay and the delay is different for every two nodes, there is a different critical set of workload traces $\mathbf{R}^{\{k\}}$ for every node k.

In the following, we present a constructive method to calculate $\mathbf{R}^{\{k\}}$. We start with calculating the critical accumulated computing time $\mathbf{Q}^{\{k\}}$ leading to T_k^*. Afterwards, we show that $R_\ell^{\{k\}}(0,t) = \Delta_\ell^A \cdot \left\lceil \frac{1}{\Delta_\ell^A} Q_\ell^{\{k\}}(0,t) \right\rceil$ is a valid workload and $\mathbf{R}^{\{k\}}(0,t) = \left[R_1^{\{k\}}(0,t), \ldots, R_n^{\{k\}}(0,t)\right]'$ actually results in the critical accumulated computing time $\mathbf{Q}^{\{k\}}$. Finally, the worst-case peak temperature $T_k^*(\tau)$ of node k at observation time τ is obtained by simulating the system with workload $\mathbf{R}^{\{k\}}(0,t)$ for all $t \in [0,\tau)$.

5.1. Critical Accumulated Computing Time

In this section, we construct the accumulated computing time $\mathbf{Q}^{\{k\}}$ that maximizes the temperature $T_k(\tau)$ at a certain observation time $\tau > 0$. In a first step, we show that each $T^*_{k,\ell}(\tau)$ defined by Eq. (10) can individually be maximized.

LEMMA 5.1 (SUPERPOSITION). *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. Then, the worst-case peak temperature of node k at time τ is:*

$$T^*_k(\tau) \leq T^{\text{init}}_k + \sum_{\ell=1}^{n} T^*_{k,\ell}(\tau) \tag{13}$$

where n is equal the number of nodes of the thermal RC circuit. Equality is obtained if the workload of all processing components is independent of each other.

Lemma 5.1 indicates that each $T_{k,\ell}(\tau)$, at a certain time instance $\tau > 0$, can individually be maximized. As $T^*_{k,\ell}$ only depends on $Q^{\{k\}}_\ell$ and $Q^{\{k\}}_\ell$ only affects $T^*_{k,\ell}$, we can individually calculate every $Q^{\{k\}}_\ell$ so that $Q^{\{k\}}_\ell$ maximizes $T_{k,\ell}(\tau)$ at time instance τ.

In the following, we introduce a method to construct the critical accumulated computing time $Q^{\{k\}}_\ell$ for each individual component ℓ. Suppose that $t^{H_{kj}}_{\max}$ denotes the time where $H_{k\ell}(t)$ is maximum. First, we will show that a higher temperature is obtained at time instance τ if the processing component is in 'active' processing mode for a longer accumulated computing time in any interval starting or ending at $\widetilde{t}^{H_{kj}}_{\max} = \tau - t^{H_{kj}}_{\max}$. For readability, we will show this statement only for $t < \widetilde{t}^{H_{kj}}_{\max}$ and refer to [Schor et al. 2012] for $t > \widetilde{t}^{H_{kj}}_{\max}$. Lemma 5.2 shows that two mode functions defined by Eq. (3) that are almost everytime the same, except in a small interval, result in different temperatures.

LEMMA 5.2 (SHIFTING). *For any given time instance τ, we consider two mode functions $S_\ell(t)$ and $\overline{S}_\ell(t)$ defined by Eq. (3) for $t \in [0, \tau)$. For given $\delta > 0, \sigma \geq 0, \sigma + 2\delta < \widetilde{t}^{H_{kj}}_{\max}$, the two mode functions only differ as follows:*

— *$S_\ell(t) = 1$ for all $t \in [\sigma, \sigma + \delta)$ ('active mode'),*
— *$S_\ell(t) = 0$ for all $t \in [\sigma + \delta, \sigma + 2\delta)$ ('idle mode'), and*
— *$\overline{S}_\ell(t) = 1 - S_\ell(t)$ for all $t \in [\sigma, \sigma + 2\delta)$.*

In other words, both mode functions have the same sequence of 'active' and 'idle' modes for $t \in [0, \sigma)$ and $t \in [\sigma + 2\delta, \tau)$. Then $\overline{T}_{k,\ell}(\tau)$ at time τ for mode function $\overline{S}_\ell(t)$ is not less than $T_{k,\ell}(\tau)$ at time τ for mode function $S_\ell(t)$, that is, $\overline{T}_{k,\ell}(\tau) \geq T_{k,\ell}(\tau)$.

Based on Lemma 5.2, we show now that a higher temperature at time τ is obtained if in any interval ending at $\widetilde{t}^{H_{kj}}_{\max}$, the processing component is in 'active' processing mode for a longer accumulated time.

LEMMA 5.3 (MODE FUNCTIONS COMPARISON). *For any given time instance τ, we consider two accumulated computing time functions Q_ℓ, resulting from mode function S_ℓ, and $\overline{Q}_\ell$, resulting from mode function $\overline{S}_\ell$, with:*

$$\overline{Q}_\ell(\widetilde{t}^{H_{kj}}_{\max} - \Delta, \widetilde{t}^{H_{kj}}_{\max}) \geq Q_\ell(\widetilde{t}^{H_{kj}}_{\max} - \Delta, \widetilde{t}^{H_{kj}}_{\max}) \tag{14}$$

for all $0 \leq \Delta \leq \widetilde{t}^{H_{kj}}_{\max}$ and $S_\ell(t) = \overline{S}_\ell(t)$ for all $\widetilde{t}^{H_{kj}}_{\max} < t \leq \tau$. Then, $\overline{T}_{k,\ell}(\tau)$ at time τ for mode function $\overline{S}_\ell(t)$ is not less than $T_{k,\ell}(\tau)$ at time τ for mode function $S_\ell(t)$.

ALGORITHM 1: Calculation of the critical accumulated computing time function $Q_\ell^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$.

Input: $b_\ell, \Delta_\ell^I, \Delta_\ell^A, \tilde{t}_{\max}^{H_{kj}}, \tau, H_{k\ell}$
Output: $Q_\ell^{\{k\}}(0,\Delta)$
for all $t^{(r)}$ in $[\tilde{t}_{\max}^{H_{kj}}, \tilde{t}_{\max}^{H_{kj}} + b_\ell - \Delta_\ell^A]$ **do**
  ▷ *find position of the burst*
  for all $t^s \in [0, \Delta_\ell^I]$ **do**
    ▷ *find gap btw burst and suc. active interval*
    $t^{(l)} = t^{(r)} - b_\ell + \Delta_\ell^A$
    $S_\ell(t) = \begin{cases} 1 & t \in [t^{(r)} - b_\ell + \Delta_\ell^A, t^{(r)}) \\ 0 & \text{otherwise} \end{cases}$
    for $i = 1$ to $\left\lceil \frac{\tau - t^{(r)}}{p_\ell} \right\rceil$ **do**
      ▷ *make trace for* $t > t^{(r)}$
      $S_\ell(t) = S_\ell(t) + v_\ell\left(t, t^s + t^{(r)} + (i-1)\cdot p_\ell\right)$
    end for
    for $i = 1$ to $\left\lceil \frac{t^{(l)}}{p_\ell} \right\rceil$ **do**
      ▷ *make trace for* $t < t^{(l)}$
      $S_\ell(t) = S_\ell(t) + v_\ell\left(t, t^s + t^{(l)} - i\cdot p_\ell\right)$
    end for
    $T_{k,\ell} = \int_0^t S_\ell(\tau) \cdot H_{k\ell}(t-\tau)\,\mathrm{d}\tau$
    if $T_{k,\ell} > T_{k,\ell}^*$ **then**
      ▷ $T_{k,\ell}$ *comparison*
      $T_{k,\ell}^* = T_{k,\ell}, \;\; S_\ell^{\{k\}} = S_\ell$
    end if
  end for
end for
$Q_\ell^{\{k\}}(0,\Delta) = \int_0^\Delta S_\ell^*(\tau)\,\mathrm{d}\tau$

Based on the previous lemmata, we will show the first main result of this section in Theorem 5.4. A constructive algorithm is introduced to calculate the critical accumulated computing time $Q_\ell^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$ that maximizes $T_{k,\ell}(\tau)$ at a certain observation time τ, that is, $Q_\ell^{\{k\}}(0,\Delta)$ leading to upper bound $T_{k,\ell}^*(\tau) \ge T_{k,\ell}(\tau)$. In particular, Algorithm 1 calculates the critical accumulated computing time $Q_\ell^{\{k\}}$ by varying both the position of the burst and the gap between burst and the first active interval. As sketched in Fig. 6, the critical accumulated computing time is the computing time that maximizes the sum of the areas below the impulse response curve where the processing component is in 'active' processing mode. b_ℓ, Δ_ℓ^I, and Δ_ℓ^A are defined as in Section 4.1 and the auxiliary function $v_\ell(t,\zeta)$ used in Algorithm 1 is one for Δ_ℓ^A time units, starting at time ζ.

THEOREM 5.4 (CRITICAL ACCUMULATED COMPUTING TIME). *Suppose that the function $v_\ell(t,\zeta)$ is defined as:*

$$v_\ell(t,\zeta) = \begin{cases} 1 & 0 \le \zeta \le t \le \min(\zeta + \Delta_\ell^A, \tau) \\ 0 & \textit{otherwise} \end{cases} \tag{15}$$

and the accumulated computing time function $Q_\ell^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$ constructed by Algorithm 1 leads to $T_{k,\ell}^*(\tau)$ at time τ. When the scheduler is work-conserving, $T_{k,\ell}^*(\tau)$ is an upper bound on the highest value of $T_{k,\ell}(\tau)$ at time τ, that is, $T_{k,\ell}^*(\tau) \ge T_{k,\ell}(\tau)$.

So far, we only derived the accumulated computing time that maximizes the temperature at observation time τ. However, we did not dwell on the amount of observation time τ. Next, we will show that increasing the observation time τ will not decrease the worst-case peak temperature T_k^* 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' mode.

LEMMA 5.5 (INITIAL TEMPERATURE). *Suppose that the accumulated computing time function $\mathbf{Q}^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$ from Theorem 5.4 leads to the worst-case peak temperature $T_k^*(\tau)$ 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 $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$.*

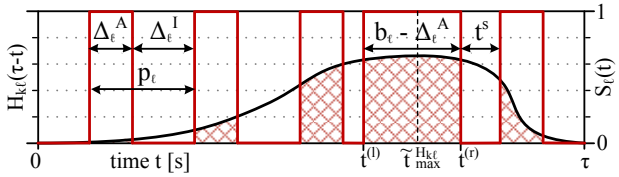


Fig. 6. Illustration of Algorithm 1 to calculate the critical accumulated computing time function $Q_\ell^{\{k\}}(0,\Delta)$.

5.2. Critical Accumulated Workload

Theorem 5.4 provides an upper bound $T^*_{k,\ell}(\tau)$ on $T_{k,\ell}(\tau)$ at a certain observation time τ, however, there might be no workload trace that leads to the critical accumulated computing time $Q_\ell^{\{k\}}(0,\Delta)$. In the following, we will show that this is not the case, and $R_\ell^{\{k\}}(0,\Delta) = \Delta_\ell^A \cdot \left\lceil \frac{1}{\Delta_\ell^A} Q_\ell^{\{k\}}(0,\Delta) \right\rceil$ actually results in the critical accumulated computing time.

THEOREM 5.6 (CRITICAL CUMULATED WORKLOAD). *Suppose that the accumulated computing time function $Q_\ell^{\{k\}}(0,\Delta)$ for all $0 \le \Delta \le \tau$ is defined as in Theorem 5.4. Then, the critical workload function $R_\ell^{\{k\}}(0,\Delta) = \Delta_\ell^A \cdot \left\lceil \frac{1}{\Delta_\ell^A} Q_\ell^{\{k\}}(0,\Delta) \right\rceil$*

— leads to the accumulated computing time $Q_\ell^{\{k\}}(0,\Delta)$, and
— complies with arrival curve α_ℓ according to Eq. (1).

Furthermore, the set of workload functions $\mathbf{R}^{\{k\}}(0,t)$

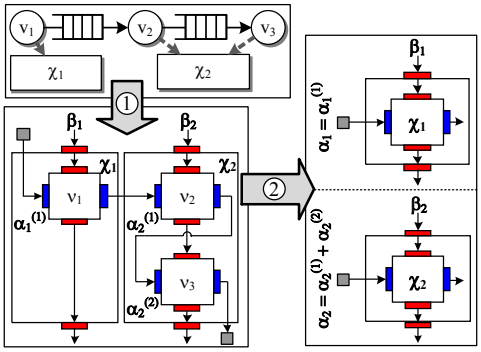
— leads to the highest possible temperature $T^*_k(\tau) \ge T_k(t)$ for all $0 \le t \le \tau$ and any set of feasible workload traces with the same initial temperature vector $\mathbf{T}^0 \le (\mathbf{T}^\infty)^i$.

Suppose that the workload of all processing components is independent of each other. Then, the upper bound $T^*_k(\tau)$ determined by Theorem 5.6 is tight as there exists a workload that leads to the critical accumulated computing time. Theorem 5.6 completed the derivation of the constructive method to calculate the worst-case peak temperature $T^*_k(\tau)$ of component k. Starting from the set of arrival curves $\boldsymbol{\alpha}(\Delta)$, we can determine $\boldsymbol{\gamma}(\Delta)$ by Eq. (2). Afterwards, Algorithm 1 constructs $Q_\ell^{\{k\}}$ for every component ℓ, which in turn determines the critical sequence of 'active' and 'idle' modes. Finally, $T^*_k(\tau)$ can be found by solving the thermal model defined by Eq. (4) at time $t = \tau$.

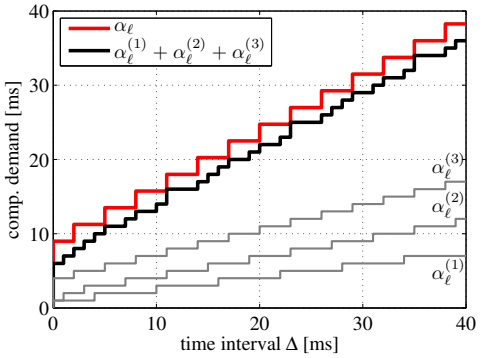
Finally note that the proposed thermal analysis method only calculates an upper bound on the maximum temperature inside the considered thermal model defined in Section 4.2. Its accuracy mainly depends on the selected granularity, that is, a higher accuracy is obtained by dividing the floorplan in cells and individually measuring the temperature of each cell [Skadron et al. 2004]. Another base assumption of our model is the linear dependency of power dissipation on temperature due to leakage power. As this assumption is only valid inside a certain temperature range [Liu et al. 2007], it is critical to select the power parameters so that they upper bound the actual power consumption in the relevant temperature range.

6. AUTOMATIC GENERATION AND CALIBRATION OF ANALYSIS MODELS

Automation is the key for fast design space exploration. Design decisions are taken on the basis of comparing different system mappings and how well they perform with respect to both timing and temperature. Integrating formal timing and thermal anal-



(a) Steps to calculate the computational demand of processing components. In particular, three processes v_1, v_2, and v_3 are assigned to two processing components χ_1 and χ_2.



(b) Calculation of the arrival curve α_ℓ for processing component ℓ that has three processes assigned. $\alpha_\ell^{(1)}$, $\alpha_\ell^{(2)}$, and $\alpha_\ell^{(3)}$ are the arrival curves of the corresponding processes.

Fig. 7. Analysis model generation.

ysis models in the system-level design space exploration loop requires both automatic generation of models and automatic calibration of these models with performance data reflecting the execution of the system on the target platform. For today's MPSoC platforms, the manual generation of analysis models and the individually calibration of all components with platform-dependent data is a major effort, which would slow down the entire design cycle and would be an actual source of errors.

In this section, we argue that for the considered class of systems, the generation of formal analysis models can be fully automated and that the analysis models can be parameterized from model calibration based on a set of benchmark mappings. We focus on the generation and calibration of the formal thermal analysis model proposed in the last section. The generation and calibration of timing analysis models is detailed in [Huang et al. 2012].

6.1. Generation of Analysis Models

The generation of a formal thermal analysis model similar to the one introduced in the last section consists in translating system specification $S = \{\mathcal{A}, \mathcal{T}, (b, \sigma)\}$ into the thermal analysis model $\Pi = (\boldsymbol{\alpha}, \Theta)$. Every arrival curve $\alpha_\ell \in \boldsymbol{\alpha}$ upper bounds the total cumulative workload of a processing component ℓ and complies to a PJD model as described in Section 4.1. Θ refers to the thermal model of the platform that is defined by the set of impulse responses represented by the matrix $\mathbf{H}$ and the power consumption vector $\mathbf{P}$. As processing components are work-conserving, the resource availability of component ℓ is implicitly abstracted by the service curve $\beta_\ell(\Delta) = \Delta$ for all intervals of length $\Delta \geq 0$.

The two-step procedure outlined in Fig. 7a is used to calculate the set of arrival curves $\boldsymbol{\alpha}$. In a first step, the MPA model $\mathcal{M} = (\mathcal{V}, E)$ of the system $S = \{\mathcal{A}, \mathcal{T}, (b, \sigma)\}$ is generated. For each process v of the process network, an MPA greedy processing component (GPC) is instantiated and for every edge in the process network, an event stream e is instantiated between the corresponding actors, by means of an MPA arrival curve. Afterwards, binding and scheduling of processes are addressed by connecting GPCs by corresponding resource streams. For example, preemptive fixed priority scheduling can be modeled by chaining the GPCs in order of their priority, and assigning full resource availability to the first GPC of the chain. In Fig. 7a, full resource

availability is assigned to the GPC corresponding to v_2, and the remaining resource availability is connected to the GPC of v_3.

In the second step, the set of arrival curves $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]'$ is calculated from the previously generated MPA model $\mathcal{M} = (\mathcal{V}, E)$. Suppose that m actors are assigned to processing component ℓ, each with arrival curve $\alpha_\ell^{(k)}$. Then the arrival curve α_ℓ is:

$$\alpha_\ell(\Delta) = \sum_{k=1}^{m} \alpha_\ell^{(k)}(\Delta). \tag{16}$$

In case that α_ℓ is not anymore complying to the PJD model, it is necessary to convert the arrival curve into the PJD model by the method presented in [Künzli et al. 2007]. As the method upper bounds the original arrival curve, the proposed thermal analysis model still leads to a safe bound on temperature. Figure 7b illustrates this step by means of a processing component that has three processes assigned.

The generation of the thermal model Θ follows directly from the platform specification. First, the coefficients $\boldsymbol{\phi}$, which reflect the temperature-dependency of the leakage power, are calculated by linearizing the power model described in [Skadron et al. 2004]. Then, the power consumption vector $\mathbf{P}$ is calculated by:

$$\mathbf{P}(t) = \boldsymbol{\phi} \cdot \left(\mathbf{T}(t) - \mathbf{T}^{\text{base}}\right) + \mathbf{P}_{\text{leakage}} + \mathbf{P}_{\text{dyn}}(t) = \boldsymbol{\phi} \cdot \mathbf{T}(t) + \boldsymbol{\psi}(t) \tag{17}$$

where $\mathbf{T}^{\text{base}}$ is the temperature vector for which the calibration parameters are obtained. The set of impulse responses is finally calculated by:

$$\mathbf{H}(t) = e^{\mathbf{A}\cdot t} \cdot \mathbf{B} = e^{-\mathbf{C}^{-1}\cdot(\mathbf{G}+\mathbf{K}-\boldsymbol{\phi})\cdot t} \cdot \mathbf{C}^{-1}. \tag{18}$$

6.2. Calibration of Analysis Models

During model generation, the timing behavior of a system is abstracted by arrival and service curves. The thermal behavior is abstracted by a constant, but temperature-dependent, power consumption and a thermal RC network. In order to obtain the required parameters for this abstraction, the application is calibrated prior to design space exploration with data corresponding to the target platform. As model calibration is specific to a certain model of computation, we restrict ourselves in this section to applications specified using an SDF representation. Table I summarizes the required parameters and categorizes them into two subsets, namely timing parameters ψ_{timing} and thermal parameters ψ_{thermal}. While timing analysis only depends upon the timing parameters ψ_{timing}, both subsets are required for the calculation of the worst-case chip temperature. Timing parameters include the worst-case and best-case execution times of a process and the characteristics of the channels. The thermal parameters include the power consumption and the thermal configuration of the platform.

Both the execution time of processes and the power consumption of processing components are associated with an uncertainty. Formal methods and tools [Wilhelm et al. 2008] are actually required to calculate hard bounds on execution time and power consumption. However, such strict formal methods have the disadvantage that the overall modeling effort drastically increases with the complexity of MPSoC platforms. An alternative for model calibration is simulation or execution on real hardware platforms. Although safe bounds cannot be guaranteed unless exhaustive test patterns are applied, they are often the only practical possibility for model calibration. Compared to real hardware platforms, simulation has the advantage that virtual platforms are often earlier available and allow non-intrusive tracking of the execution. Therefore, in the next section, the prototype implementation of the mapping optimization frame-

Table I. Model parameters required by MPA. The parameters are categorized into two subsets, namely timing parameters ψ_{timing} and thermal parameters ψ_{thermal}.

Entity	Parameter	Unit	Source	Category
process v	worst-case execution time $\text{WCET}(v)$ / best-case execution time $\text{BCET}(v)$	sec / iteration	low-level sim.	time
queue q	min. / max. token size $N_{\min}(q)$, $N_{\max}(q)$	bytes / access	functional sim.	time
	write rate p_q, read rate c_q	1	functional sim.	time
processor χ	clock frequency $f(\chi)$	cycles / sec	hardware data-sheet	time
	dynamic power consumption $\overline{P}_{\text{dyn}}(\chi)$	W	low-level sim.	thermal
	leakage power consumption $\overline{P}_{\text{leakage}}(\chi)$	W	low-level sim.	thermal
architecture $\mathcal{T}$	capacitance matrix $\mathbf{C}$	J/K	low-level sim.	thermal
	conductance matrix $\mathbf{G}$ / ground conductance matrix $\mathbf{K}$	W/K	low-level sim.	thermal

work uses simulation on a cycle-accurate virtual platform to determine worst-case execution times and power consumptions.

Time-independent parameters, that is, the token consumption behavior of the channels, are determined by monitoring the `read` and `write` calls during functional simulation. Time-dependent parameters, namely the best-case and worst-case execution times of processes are obtained by monitoring the system in a cycle-accurate simulator. After tracking all context switches, the individual execution times of processes are calculated by means of a log-file analysis, where the collected information is first decoded and then stored in a database.

In order to predict the worst-case chip temperature, thermal parameters have to be determined that provide safe bounds on power dissipation. The set of thermal parameters ψ_{thermal} is composed of two assignments, as described in Table I:

$$\psi_{\text{thermal}} : \begin{cases} \chi \mapsto \left\{\overline{P}_{\text{leakage}}(\chi), \overline{P}_{\text{dyn}}(\chi)\right\} & \forall \chi \in \mathcal{T} \\ \mathcal{T} \mapsto \{\mathbf{C}, \mathbf{G}, \mathbf{K}\}. \end{cases} \tag{19}$$

A possible method to determine the maximum power consumption of a processing component is to sum up the power dissipations of all individual units, according to data provided by hardware designers in data sheets. However, it is unlikely that all units are simultaneously active and the obtained power numbers would drastically outperform the actual power consumption. Therefore, we use, again, a simulation-based approach to obtain better estimates of power consumptions.

The power parameters of each processing component are obtained by monitoring separately the dynamic and leakage power dissipation in a cycle-accurate simulator. As the thermal model assumes time-independent leakage power, $\overline{P}_{\text{leakage}}(\chi)$ is set to the maximum value of the leakage power that is observed for processing component χ during execution. A multi-step procedure is required to determine the dynamic power consumption. First, the transient dynamic power consumption of each processing component is recorded as power traces. Second, the power traces are divided into individual processes so that the average dynamic power consumption $\overline{P}^{v,i}_{\text{dyn}}$ of process v in iteration i is determined as:

$$\overline{P}^{v,i}_{\text{dyn}} = \frac{1}{t^{v,i}_{\text{end}} - t^{v,i}_{\text{start}}} \int_{t^{v,i}_{\text{start}}}^{t^{v,i}_{\text{end}}} P^{v}_{\text{dyn}}(t)\,\mathrm{d}t \tag{20}$$

where $t^{v,i}_{\text{start}}$ and $t^{v,i}_{\text{end}}$ are the start and end time of iteration i, respectively. Then, we use the maximum average value of $P_{\text{dyn}}(t)$ as dynamic power consumption $\overline{P}_{\text{dyn}}(\chi)$:

$$\overline{P}_{\text{dyn}}(\chi) = \max_{\forall v \text{ assigned to } \chi} \left(\max_{\forall i} \left(\overline{P}^{v,i}_{\text{dyn}} \right) \right). \tag{21}$$

The thermal matrices $\mathbf{G}$, $\mathbf{S}$, and $\mathbf{C}$ are computed by means of the method described in [Skadron et al. 2004]. The parameters required for this calculation, in particular the three-dimensional floorplan and the thermal configuration of the chip, are known from architecture specification.

In our mapping optimization framework, model calibration is performed just once in the design flow, before design space exploration. As all functional parameters are mapping-independent, they only have to be obtained once for all candidate mappings. The same applies to the thermal configuration of the platform, that is, the $\mathbf{G}$, $\mathbf{K}$, and $\mathbf{C}$ matrices. All other parameters obtained by cycle-accurate simulation vary for each candidate mapping, but can be estimated by a set of benchmark mappings that covers all possible configurations. In the following experiments, we use the MPARM [Benini et al. 2005] virtual platform for cycle-accurate simulation. As MPARM uses direct memory access controllers to reduce the interference between computation and communication, and all processing components are of the same type, the number of benchmark mappings can be reduced to one.

7. EXPERIMENTS

In this section, a prototype implementation of the proposed system-level mapping optimization framework is used to explore designs with respect to both worst-case timing parameters and chip temperature. In the first case study, the viability of the proposed thermal analysis method is discussed by comparing the transient temperature evolution for three different evaluation methods. Then, in the second case study, we evaluate the time to perform design space exploration and the quality of the obtained temperature bounds. Finally, the design space of two benchmark applications is explored and Pareto optimal candidate mappings are identified in the third case study. The MPARM [Benini et al. 2005] cycle-accurate simulator is used in all experiments.

7.1. Experimental Setup

A homogeneous multi-processor ARM platform with three processing components connected via a shared bus is considered as target platform. The MPA framework [Wandeler et al. 2006] has been extended with the ability to calculate the worst-case chip temperature by the method proposed in Section 5 and integrated into the prototype implementation of our mapping optimization framework to calculate the worst-case chip temperature and the overall latency of different candidate mappings.

Automated model calibration is performed based on timing and power numbers extracted from MPARM virtual platform and thermal parameters from HotSpot [Huang et al. 2006], see Table II. In order to reduce the dimensionality of the search space, fixed priority preemptive scheduling is used on all processors while a TDMA policy is employed on the bus. However, our framework can easily be extended to consider other scheduling and arbitration policies, if they are available in the final system. All experiments have been performed on a $3.20\,\text{GHz}$ Intel Pentium D machine with $2\,\text{GB}$ of RAM.

7.2. Case Study Applications

In this section, we analyze the timing and temperature behavior of four benchmark applications:

Table II. Thermal configuration of HotSpot.

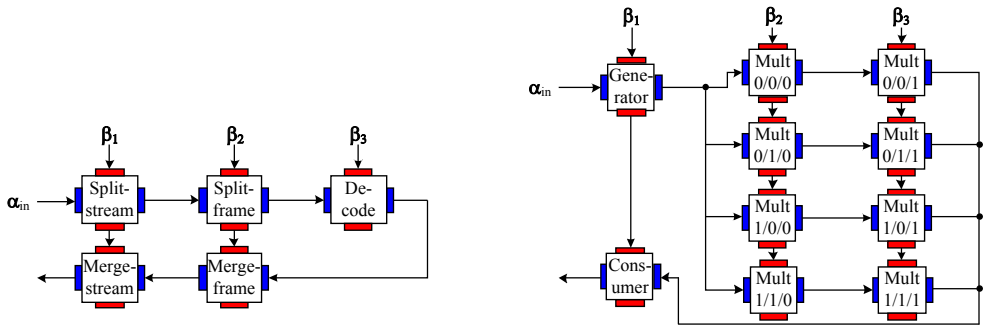
Parameter	Symbol	Value
Silicon thermal conductance $[\mathrm{W}/(\mathrm{m}\cdot\mathrm{K})]$	k_{chip}	150
Silicon specific heat $[\mathrm{J}/(\mathrm{m}^3\cdot\mathrm{K})]$	p_{chip}	$1.75\cdot10^6$
Thickness of the chip $[\mathrm{mm}]$	t_{chip}	3.5
Convection resistance $[\mathrm{K}/\mathrm{W}]$	r_{convec}	2
Heatsink width $[\mathrm{m}]$	s_{sink}	0.017
Heatsink thickness $[\mathrm{mm}]$	t_{sink}	0.01
Heatsink thermal conductance $[\mathrm{W}/(\mathrm{m}\cdot\mathrm{K})]$	k_{sink}	400
Heatsink specific heat $[\mathrm{J}/(\mathrm{m}^3\cdot\mathrm{K})]$	p_{sink}	$3.55\cdot10^6$
Thickness of the interface material $[\mathrm{mm}]$	$t_{\text{interface}}$	0.04
Ambient temperature $[\mathrm{K}]$	T_{amb}	300

Producer-Consumer (P-C): The P-C example is a simple application that consists of five pipelined, parallel processes. The producer generates a stream of floating-point numbers, which are passed to the first worker process. After computing a few arithmetic operations, each worker process forwards the floating-point number to the next worker process until the consumer receives it.

Motion-JPEG (MJPEG) decoder: The MJPEG decoder is a video codec in which each video frame is separately compressed as a JPEG image. The MPA model of the considered configuration is outlined in Fig. 8a. The decoder's input (abstracted by the arrival curve α_{in}) is a video stream that is first split into individual frames. The second actor splits the frames into macroblocks to separately decode each macroblock. Afterwards, the decoded macroblocks are stitched together into a frame and finally into a stream.

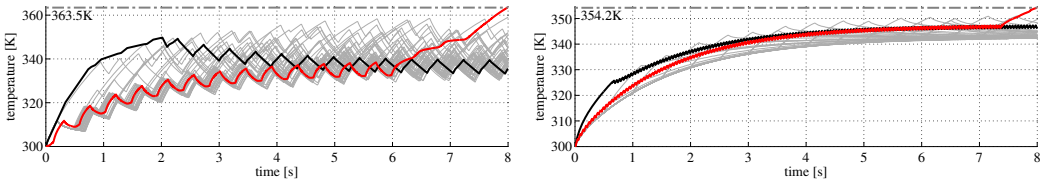
Fast Fourier transform (FFT): In order to compute an 8-point FFT, a distributed $\log(8)$-stage butterfly network has been implemented [Wu and Chang 1993]. Each stage is composed of four multiply-subtract-add modules so that the application can be split up into twelve processes, each computing a multiply-subtract-add module.

Matrix multiplication: The distributed matrix multiplication application computes the product of two $N \times N$ matrices by subdividing the matrix product into single multiplications and additions. This way, the application is split up into single processes, each performing a multiplication followed by an addition. To feed the structure and to retrieve the result, an input generator and an output consumer are added. Figure 8b shows the MPA model of the considered configuration for $N = 2$.



(a) MJPEG decoder. (b) Matrix multiplication.

Fig. 8. Component model of the considered example applications. For the sake of simplicity, the component model of the bus is not shown. α_{in} abstracts the system's input stream. β_1, β_2, and β_3 abstract the available resources of component χ_1, χ_2, and χ_3, respectively.



(a) MJPEG decoder, processing component 3. (b) Matrix multiplication, processing component 1.

Fig. 9. Temperature evolution of the node corresponding to the processing component with the maximum temperature when the system is processing the MJPEG decoder or the matrix multiplication application.

7.3. Peak Temperature Analysis

First, we study the viability of the thermal analysis method proposed in Section 5. To this end, we compare the transient temperature evolution and the resulting maximum temperature for different evaluation methods. In particular, the temperature evolution of the thermal critical instance is compared to the temperature of the timing critical instance, and to the temperature of 40 workload traces simulated on a cycle-accurate simulation tool-chain based on MPARM and HotSpot. The thermal critical instance is the trace that leads to the worst-case chip temperature and the timing critical instance is the trace that releases the workload as early as possible. In order to stress the system, the 40 input traces executed on the cycle-accurate simulator cover the whole range of possible input streams. In particular, among the 40 traces, there are input streams where the number of concurrently arriving events equals the maximum length of the input stream's burst.

The set-up of the cycle-accurate thermal simulation tool-chain is detailed in [Thiele et al. 2011]. Note that the tool-chain is similar to other well-established thermal-aware simulation platforms, as for example [Bartolini et al. 2010; Garcia del Valle and Atienza 2010] except that a different target platform is used to obtain power values. Figure 9 presents the results of the transient temperature simulation in the interval $[0, 8\,]s$ when the observation time τ is set to eight seconds. Even though we only show the temperature evaluation of two benchmark applications, the findings for the other benchmark applications show similar trends.

The maximum temperature caused by the timing critical instance is $349.8\,\mathrm{K}$ for the MJPEG decoder and $347.3\,\mathrm{K}$ for the matrix multiplication. As it releases the workload as early as possible, the timing critical instance heats up the system faster than any other trace at the beginning, but the system cools down to a later point in time. All other traces that are recorded on the cycle-accurate simulator heat up the chip the most when multiple events arrive concurrently. In particular, the average maximum temperature caused by these workload traces is $352.4\,\mathrm{K}$ and $346.1\,\mathrm{K}$, and the highest maximum temperature is $360.1\,\mathrm{K}$ and $351.1\,\mathrm{K}$ for the MJPEG decoder and the matrix multiplication, respectively. The thermal critical instance places all bursts close to the observation time τ so that at time $\tau = 8\,\mathrm{s}$, a maximum temperature of $363.5\,\mathrm{K}$ is observed for the MJPEG decoder. Similarly, the worst-case chip temperature of the matrix multiplication is $354.2\,\mathrm{K}$. The average durations for simulating the transient temperature evolutions are shown in Table III. Simulating the temperature evolution on the cycle-accurate simulator is on average 266.2 times slower than calculating the thermal critical instance.

Workload traces, which have an input stream where the number of concurrently arriving events equals the maximum length of the input stream's burst, lead to a high maximum temperature that might only be a few degrees below the worst-case chip temperature. In particular, the difference between the maximum temperature of such

Table III. Duration for simulating the transient temperature evolution for different evaluation methods.

	P-C	MJPEG	FFT	Matrix
Thermal critical instance	84.8 s	94.3 s	83.0 s	90.5 s
Timing critical instance	1.8 s	2.2 s	2.2 s	2.2 s
Cycle-accurate simulator	24876 s	22428 s	22873 s	23372 s

workload traces and the worst-case chip temperature may be caused due to several reasons. First, the power consumption depends on various impact factors like cache misses and bus congestions. Second, as shown in Section 5, the worst-case chip temperature results if all cores simultaneously process a specific pattern, which might be different from a bursty input stream.

7.4. Efficiency and Quality

After discussing the viability of the thermal analysis method, we integrate it into the mapping optimization framework and measure its efficiency and quality. To this end, we first obtain the model parameters from executing the application with a benchmark mapping. Afterwards, during design space exploration, these model parameters are used in different configurations to individually analyze every candidate mapping. To measure the efficiency, we compare the duration of analysis model generation, calibration, and evaluation. In order to evaluate the quality of the obtained results, the worst-case chip temperature calculated by MPA is compared to the maximum temperature observed on a ten seconds system simulation using the previously described cycle-accurate simulation tool-chain, and the maximum steady-state temperature calculated out of the average power consumption [Skadron et al. 2004]. Both methods correlate with the typical state-of-the-art thermal evaluation methods used in thermal aware task allocation and scheduling algorithms [Chantem et al. 2008; Xie and Hung 2006].

The duration of analysis model generation, calibration, and evaluation are listed in Table IV. First, we note that calibration is two to three orders of magnitude slower than model generation and evaluation. Out of the three steps to perform model calibration, cycle-accurate simulation on MPARM is the most time consuming step. The duration of the log-file analysis mainly depends on the number of context switches, which in turn depends on the number of processes. As a new log entry is created for every context switch, the duration of the log-file analysis is long for the FFT application, where many dependent processes are concurrently executed. Furthermore, both the duration of model calibration and the accuracy of the obtained results are affected by the length of the execution trace. In general, longer execution traces increase the

Table IV. Duration of analysis model generation, calibration, and evaluation.

		P-C	MJPEG	FFT	Matrix
Model calibration (one-time effort)	Synthesis (incl. func. simulation)	37 s	58 s	47 s	39 s
	Simulation on MPARM	24709 s	22504 s	22752 s	23510 s
	Log-file analysis	114 s	114 s	1847 s	292 s
	Overall time for one mapping	24861 s	22675 s	24646 s	23842 s
Design space exploration	Model generation	2 s	3 s	2 s	2 s
	Model evaluation	96 s	132 s	96 s	119 s
	Overall time for one mapping	98 s	135 s	98 s	121 s

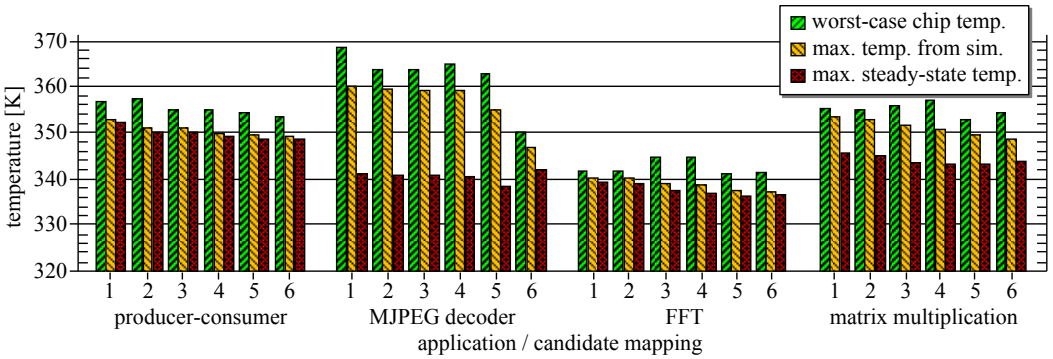


Fig. 10. Comparison of the worst-case chip temperature calculated by MPA, the maximum temperature observed on a ten seconds system simulation, and the maximum steady-state temperature for six candidate mappings per benchmark application.

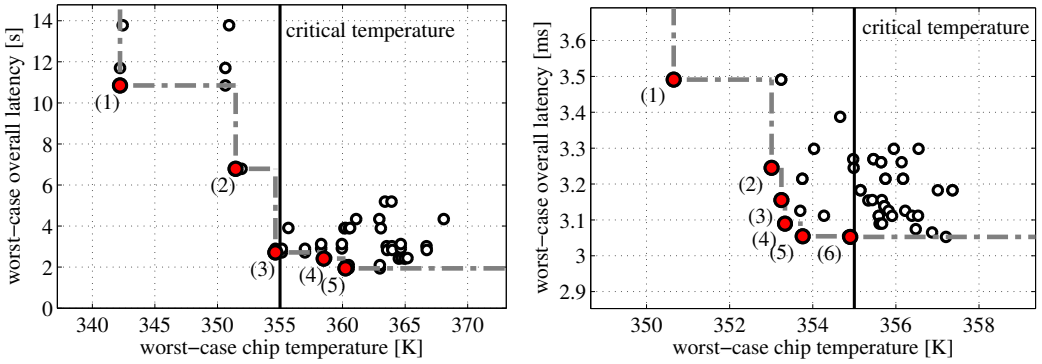
calibration time but result in better calibration data with respect to worst-case execution time and maximum average power consumption.

In Fig. 10, the worst-case chip temperature calculated by MPA is compared with the maximum temperature observed on a ten seconds system simulation using the cycle-accurate simulation tool-chain and with the maximum steady-state temperature. Totally, we evaluated six randomly selected candidate mappings per benchmark application. The values are ordered by the maximum temperature of the system simulation in descending order, per benchmark application.

First, we note that the worst-case chip temperature always upper bounds the maximum temperature obtained from cycle-accurate simulation. As cycle-accurate simulation is several orders of magnitude slower than formal thermal analysis, this confirms our approach to include formal worst-case analysis models in the design space loop and to use cycle-accurate simulation only once for model calibration. Second, one can draw the conclusion that the steady-state temperature is no indicator for a low worst-case chip temperature. Third, we note that a mapping candidate with a lower maximum temperature from simulation does not necessary have a lower worst-case chip temperature. On the one hand, 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. On the other hand, the proposed thermal analysis method does not lead to a tight bound on the maximum temperature of the system, due to several reasons:

— The parameters modeling the power consumption are the maximum average power consumption per iteration, thus overestimating the actual power consumption.
— The workload curves of the processing components are not independent of each other as the component model results from a single process network.
— As not all arrival curves comply with a PJD model, original arrival curves have to be upper bounded.

Unlike simulation, the proposed thermal analysis method calculates a safe bound on the maximum temperature of a candidate mapping. The difference between the worst-case chip temperature and the maximum temperature of the system simulation is the worst possible inaccuracy of the formal worst-case analysis method. Nonetheless, due to the over-approximation of a tight bound on the maximum temperature of the system, another candidate mapping might actually have a lower maximum temperature. To avoid the selection of a wrong candidate mapping, the designer might keep more



(a) MJPEG decoder application. (b) Matrix multiplication application.

Fig. 11. Worst-case chip temperature vs. latency for 50 mapping configurations.

than one mapping candidate after the first design space exploration, and reevaluates these candidates with either a higher accuracy or a cycle-accurate simulator.

In summary, calculating the worst-case chip temperature by means of a formal analysis model is desirable. First, identifying the worst-case chip temperature is notably important in real-time systems. The functional correctness of a processor is often only given as long as a certain critical temperature is not exceeded. Therefore, thermal aware mapping optimization algorithms have to include in their objective function the worst-case chip temperature. Second, as the formal thermal analysis method is several orders of magnitude faster than cycle-accurate simulation, it enables a much faster and more complete exploration of the design space.

7.5. Design Space Exploration

In the third case study, we use a prototype implementation of our mapping optimization framework to explore parts of the MJPEG decoder's and the matrix multiplication's design space. The design space is formed by two analysis metrics, namely the worst-case chip temperature and the overall latency of the application, both calculated by the extended MPA framework.

Figure 11 shows the scatter plots for the two benchmark applications having the worst-case chip temperature on the horizontal axis and the latency on the vertical axis. Each of the 50 points refers to a different candidate mapping of the benchmark applications. It shows that no solution is optimal with respect to both latency and temperature. For the MJPEG decoder, the worst-case chip temperature of the candidate mapping leading to the lowest latency, that is, $1.94\,\mathrm{s}$, is $360.2\,\mathrm{K}$ and the candidate mapping leading to the lowest worst-case chip temperature of $342.2\,\mathrm{K}$ has a worst-case latency of $11.7\,\mathrm{s}$. Similarly, for the matrix multiplication, the worst-case chip temperature of the candidate mapping leading to the lowest latency is $354.9\,\mathrm{K}$, but the lowest worst-case chip temperature is just $350.6\,\mathrm{K}$. The Pareto-optimal candidates are highlighted in the figure. In particular, there are five Pareto-optimal candidates for the MJPEG decoder and six Pareto-optimal candidates for the matrix multiplication. For example, suppose that the target platform has a critical temperature of $355\,\mathrm{K}$, that is, exceeding this temperature might lead to functional errors. Therefore, the designer would select the mapping with the lowest latency from all mappings that have a worst-case chip temperature smaller than $355\,\mathrm{K}$. Programmed to consider this temperature constraint, the mapping optimization framework would select mapping (3) as the optimal mapping of the system for the MJPEG decoder and mapping (6) as optimal mapping

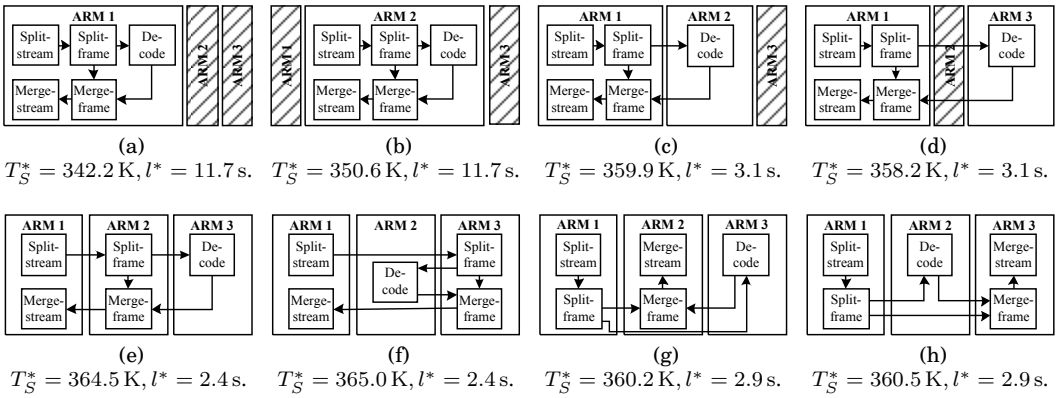


Fig. 12. Eight mapping configurations of the MJPEG decoder application together with their worst-case chip temperature T_S^* and their latency l^*.

for the matrix multiplication. As the proposed algorithm offers safe bounds, the system can safely execute the mapping without further involving other (dynamic) thermal management strategies, which may lead to unpredictable behavior.

To study the connection between the worst-case chip temperature and worst-case latency in greater detail, Fig. 12 outlines eight selected mapping configurations of the MJPEG decoder application together with their worst-case chip temperature T_S^* and their latency l^*. In order to study the effect of the placement, we study solution pairs where only the placement of the processing components has changed. For example, mappings 12a and 12b, and mappings 12c and 12d are solution pairs. It shows that the physical placement cannot be ignored in temperature analysis. For example, mappings 12a and 12b have almost the same latency, but their peak temperatures differ by more than $8\,\mathrm{K}$. Therefore, even if the mapping is already predefined, the system designer might reduce the temperature by selecting an appropriate physical placement.

8. CONCLUSIONS

In this article, a high-level optimization framework for mapping real-time applications onto embedded MPSoC architectures is presented that guarantees the final performance and correct function of the system, considering both functional and non-functional properties. As high chip temperatures may lead to long-term reliability concerns and short-term functional errors, providing guarantees on maximum temperature is as important as functional correctness and timeliness for embedded real-time systems. In order to include formal timing and thermal analysis in design space exploration, the analysis models are automatically generated from the same set of specifications as used for software synthesis. Afterwards, the analysis models are automatically calibrated with performance data reflecting the execution of the system on the target platform. The data is automatically obtained prior to design space exploration based on a set of benchmark mappings. The considered thermal analysis model is able to address various thermal effects like the heat exchange between neighboring processing components and temperature-dependent leakage power to accurately model the thermal behavior of modern MPSoC architectures. Finally, the high-level mapping optimization framework is applied to a multiprocessor ARM platform to validate the effectiveness of the approach based on four benchmark applications. It shows that optimizing a system with respect to both performance and temperature will lead to major benefits.

REFERENCES

ATIENZA, D. ET AL. 2007. HW-SW Emulation Framework for Temperature-Aware Design in MPSoCs. *ACM Trans. Des. Autom. Electron. Syst. 12,* 3, 1–26.

BACIVAROV, I., HAID, W., HUANG, K., AND THIELE, L. 2010. Methods and Tools for Mapping Process Networks onto Multi-Processor Systems-On-Chip. In *Handbook of Signal Processing Systems*, S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, Eds. Springer, New York, NY, USA, 1007–1040.

BARTOLINI, A., CACCIARI, M., TILLI, A., BENINI, L., AND 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)*. ACM, Providence, Rhode Island, USA, 311–316.

BENINI, L., BERTOZZI, D., BOGLIOLO, A., MENICHELLI, F., AND OLIVIERI, M. 2005. MPARM: Exploring the Multi-Processor SoC Design Space with SystemC. *J. VLSI Signal Process 41,* 2, 169–182.

BIRKHOFF, G. AND VARGA, R. S. 1958. Reactor Criticality and Nonnegative Matrices. *J. Soc. Ind. Appl. Math. 6,* 4, 354–377.

BROOKS, D. AND MARTONOSI, M. 2001. Dynamic Thermal Management for High-Performance Microprocessors. In *Proc. Int'l Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Monterrey , Mexico, 171–182.

CHANTEM, T., DICK, R. P., AND HU, X. S. 2008. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *Proc. Design, Automation and Test in Europe (DATE)*. ACM/IEEE, Munich, Germany.

COSKUN, A. K., ROSING, T. S., AND WHISNANT, K. 2007. Temperature Aware Task Scheduling in MPSoCs. In *Proc. Design, Automation and Test in Europe (DATE)*. ACM/IEEE, Nice, France.

DONALD, J. AND MARTONOSI, M. 2006. Techniques for Multicore Thermal Management: Classification and New Exploration. In *Proc. Int'l Symposium on Computer Architecture (ISCA)*. IEEE, Boston, USA.

FRIEDLAND, B. 1986. *Control Systems Design: An Introduction to State-Space Methods*. McGraw-Hill, New York, NY, USA.

GARCIA DEL VALLE, P. AND ATIENZA, D. 2010. Emulation-Based Transient Thermal Modeling of 2D/3D Systems-on-Chip with Active Cooling. *Microelectronics Journal 41,* 10, 1–9.

GUPTA, R. ET AL. 1997. The Elmore Delay as a Bound for RC Trees with Generalized Input Signals. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst 16*, 95–104.

HARDAVELLAS, N., FERDMAN, M., FALSAFI, B., AND AILAMAKI, A. 2011. Toward Dark Silicon in Servers. *IEEE Micro 31,* 4, 6–15.

HENIA, R., HAMANN, A., JERSAK, M., RACU, R., RICHTER, K., AND ERNST, R. 2005. System Level Performance Analysis - The SymTA/S Approach. *IEEE Proc. Computers and Digital Techniques 152,* 2, 148–166.

HUANG, K., HAID, W., BACIVAROV, I., KELLER, M., AND THIELE, L. 2012. Embedding Formal Performance Analysis into the Design Cycle of MPSoCs for Real-time Streaming Applications. *ACM Trans. Embedd. Comput. Syst. 11,* 1, 8:1–8:23.

HUANG, W., GHOSH, S., VELUSAMY, S., SANKARANARAYANAN, K., SKADRON, K., AND 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.

KAHLE, J. A., DAY, M. N., HOFSTEE, H. P., JOHNS, C. R., MAEURER, T. R., AND SHIPPY, D. 2005. Introduction to the Cell Multiprocessor. *IBM Journal of Research and Development 49,* 4.5, 589 –604.

KANGAS, T., KUKKALA, P., ORSILA, H., SALMINEN, E., HÄNNIKÄINEN, M., HÄMÄLÄINEN, T. D., RIIHIMÄKI, J., AND KUUSILINNA, K. 2006. UML-Based Multiprocessor SoC Design Framework. *ACM Trans. Embedd. Comput. Syst. 5*, 281–320.

KIENHUIS, B., DEPRETTERE, E., VISSERS, K., AND VAN DER WOLF, P. 1997. An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures. In *Proc. Int'l Conf. on Application-Specific Systems, Architectures and Processors (ASAP)*. IEEE, Zurich , Switzerland, 338–349.

KRUM, A. 2000. Thermal Management. In *The CRC Handbook of Thermal Engineering*, F. Kreith, Ed. CRC Press, Boca Raton, FL, USA, 1–92.

KUMAR, A., SHANG, L., PEH, L., AND JHA, N. 2006. HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management. In *Proc. Design Automation Conference (DAC)*. ACM, San Francisco, CA, USA, 548–553.

KÜNZLI, S., HAMANN, A., ERNST, R., AND 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)*. ACM, Salzburg, Austria.

LE BOUDEC, J.-Y. AND THIRAN, P. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Lecture Notes In Computer Science Series, vol. 2050. Springer, Berlin, Heidelberg.

LEE, E. AND MESSERSCHMITT, D. 1987. Synchronous Data Flow. *Proc. IEEE 75,* 9, 1235–1245.

LIU, Y. ET AL. 2007. Accurate Temperature-Dependent Integrated Circuit Leakage Power Estimation is Easy. In *Proc. Design, Automation and Test in Europe (DATE)*. ACM/IEEE, Nice, France.

LOH, G. H. 2008. 3D-Stacked Memory Architectures for Multi-core Processors. In *Proc. Int'l Symposium on Computer Architecture (ISCA)*. IEEE, Beijing, China, 453–464.

MAEDA, H., KODAMA, S., AND KAJIYA, F. 1977. Compartmental System Analysis: Realization of a Class of Linear Systems with Physical Constraints. *IEEE Trans. Circuits and Systems 24,* 1, 8–14.

PAOLUCCI, P. S., JERRAYA, A. A., LEUPERS, R., THIELE, L., AND VICINI, P. 2006. SHAPES:: A Tiled Scalable Software Hardware Architecture Platform for Embedded Systems. In *Proc. Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. ACM, Seoul, Korea, 167–172.

PIMENTEL, A. 2008. The Artemis Workbench for System-Level Performance Evaluation of Embedded Systems. *Int'l Journal on Embedded Systems 3,* 3, 181–196.

RAI, D., YANG, H., BACIVAROV, I., CHEN, J.-J., AND THIELE, L. 2011. Worst-Case Temperature Analysis for Real-Time Systems. In *Proc. Design, Automation and Test in Europe (DATE)*. ACM/IEEE, Grenoble, France.

SCHOR, L., BACIVAROV, I., YANG, H., AND THIELE, L. 2012. Worst-Case Temperature Guarantees for Real-Time Applications on Multi-Core Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, Beijing, China.

SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-Aware Microarchitecture: Modeling and Implementation. *ACM Trans. Architec. Code Optim. 1,* 1, 94–125.

SRIDHAR, M., RAJ, A., VINCENZI, A., RUGGIERO, M., BRUNSCHWILER, T., AND 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)*. ACM/IEEE, San Jose, CA, USA, 463–470.

THIELE, L., BACIVAROV, I., HAID, W., AND HUANG, K. 2007. Mapping Applications to Tiled Multiprocessor Embedded Systems. In *Proc. Int'l Conf. on Application of Concurrency to System Design (ACSD)*. IEEE, Bratislava, Slovak Republic, 29–40.

THIELE, L., CHAKRABORTY, S., AND NAEDELE, M. 2000. Real-Time Calculus for Scheduling Hard Real-Time Systems. In *Proc. IEEE Int'l Symposium on Circuits and Systems (ISCAS)*. Vol. 4. IEEE, Geneva, Switzerland, 101–104.

THIELE, L., SCHOR, L., YANG, H., AND BACIVAROV, I. 2011. Thermal-Aware System Analysis and Software Synthesis for Embedded Multi-Processors. In *Proc. Design Automation Conference (DAC)*. ACM, San Diego, California, 268–273.

WANDELER, E., THIELE, L., VERHOEF, M., AND LIEVERSE, P. 2006. System Architecture Evaluation Using Modular Performance Analysis: A Case Study. *Int'l Journal on Software Tools for Technology Transfer 8,* 6, 649–667.

WATANABE, Y., DAVIS, J. D., AND WOOD, D. A. 2010. WiDGET: Wisconsin Decoupled Grid Execution Tiles. In *Proc. Int'l Symposium on Computer Architecture (ISCA)*. ACM, Saint-Malo, France, 2–13.

WILHELM, R. ET AL. 2008. The Worst-Case Execution-Time Problem – Overview of Methods and Survey of Tools. *ACM Trans. Embedd. Comput. Syst. 7*, 1–53.

WU, C.-W. AND CHANG, C.-T. 1993. FFT Butterfly Network Design for Easy Testing. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 40,* 2, 110–115.

XIE, Y. AND HUNG, W.-L. 2006. Temperature-Aware Task Allocation and Scheduling for Embedded Multiprocessor Systems-on-Chip (MPSoC) Design. *The Journal of VLSI Signal Processing 45,* 3, 177–189.

YANG, C.-Y., CHEN, J.-J., THIELE, L., AND KUO, T.-W. 2010. Energy-Efficient Real-Time Task Scheduling with Temperature-Dependent Leakage. In *Proc. Design, Automation and Test in Europe (DATE)*. ACM/IEEE, Dresden, Germany, 9–14.

ZHAO, M., CHILDERS, B. R., AND SOFFA, M. L. 2005. A Model-Based Framework: An Approach for Profit-Driven Optimization. In *Proc. Int'l Symposium on Code Generation and Optimization (CGO)*. ACM/IEEE, San Jose, California, 317–327.

ZHU, C., GU, Z., SHANG, L., DICK, R., AND JOSEPH, R. 2008. Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst 27,* 8, 1479–1492.

Received November 2011; revised March 2012; accepted June 2012