Thermal-Aware System Analysis and Software Synthesis for Embedded Multi-Processors

Lothar Thiele, Lars Schor, Hoeseok Yang, Iuliana Bacivarov
Computer Engineering and Networks Laboratory
ETH Zurich, 8092 Zurich, Switzerland
firstname.lastname@tik.ee.ethz.ch

ABSTRACT

Nowadays, the reliability and performance of modern embedded multi-processor systems is threaten by the ever-increasing power densities in integrated circuits, and a new additional goal of software synthesis is to reduce the peak temperature of the system. However, in order to perform thermal-aware mapping optimization, the timing and thermal characteristics of every candidate mapping have to be analyzed. While the task of analyzing timing characteristics of design alternatives has been extensively investigated in recent years, there is still a lack of methods for accurate and fast thermal analysis. In order to obtain desired evaluation times, the system has to be simulated at a high abstraction level. This often results in a loss of accuracy, mainly due to missing knowledge of system's characteristics. This paper addresses this challenge and presents methods to automatically calibrate high-level thermal evaluation methods. Furthermore, the viability of the methods for automated model calibration is illustrated by means of a novel high-level thermal evaluation method.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Realtime and embedded systems

General Terms

Design, Performance

Keywords

Multi-Processor Systems-on-Chip, Design Space Exploration, Thermal-Aware Analysis, Model Calibration

1. INTRODUCTION

Current trends in microprocessor design include the integration of multiple processors on a single chip in order to keep pace with the ever-increasing demand of computation performance. However, the obtained increase in performance imposes a major rise in power density, which in turn threats the reliability and the performance of a system. At the same time, sufficient cooling becomes infeasible in various embedded systems like cell phones. One way to address these challenges is to optimize the system for peak temperature during software synthesis. This requires filtering out design alternatives that do not conform to the thermal requirements of the system already in early design stages, at system-level. Consequently, new system-level methodologies for analyzing both performance and thermal characteristics of design alternatives must be established in order to enable thermal-aware mapping optimization of embedded multi-processor systems.

The de-facto standard for thermal analysis is nowadays cycle-accurate simulation including temperature analysis [21], which is limited in performance and consequently, not suited for mapping optimization. The evaluation time can be reduced by increasing the abstraction level of thermal analysis. However, one of the main criticisms of system-level methods is a reduced complexity that results in a loss of accuracy, mainly due to missing knowledge about the implementation details. To overcome this problem, we propose a systematic approach to automatically calibrate system-level simulation models for thermal evaluation. Moreover, we argue in this paper that the calibration of high-level thermal evaluation models can be automated completely.

To illustrate our approach for automated model calibration, a system-level simulation model for thermal evaluation is introduced, that is based on the same specifications that are used for system software synthesis. The contributions of this paper can be summarized as follows:

- A systematic approach to integrate thermal analysis into software synthesis for multi-processor systems is developed.
- It is shown that the required system-level thermal model parameters can automatically be obtained from functional and low-level simulation.
- The viability of the proposed calibration and simulation methods is illustrated by means of a prototype implementation in DOL [22], based on the MPARM virtual platform [4].

The remainder of the paper is organized as follows: First, a review of thermal analysis and software synthesis is given. Afterwards, Section 3 presents the tackled problem and the proposed approaches. In Section 4, the considered problem is formally defined. Section 5 discusses a method to calculate thermal characteristics of a multi-processor streaming application. Section 6 describes the automated model calibration. Section 7 illustrates the proposed approaches in a case study, and finally, Section 8 concludes the paper.

DAC 2011, June 5-10, 2011, San Diego, California, USA.

2. REVIEW OF THERMAL ANALYSIS AND SOFTWARE SYNTHESIS

Due to increasing power densities in modern integrated circuits, thermal management methods have been a hot topic in research in recent years. For example, in [19], a convex optimization technique for temperature-aware frequency assignment is proposed. Various architectural-level techniques for thermal management like dynamic voltage and frequency scaling (DVFS) and the stop-go scheduling policy are compared in [8]. Coskun et al. evaluated various thermal-aware job scheduling techniques in [6, 7], and, in [25], ThreshHot, a technique to reduce the number of thermal trespasses by selecting at each step the hottest job that does not exceed the thermal threshold, is proposed.

However, thermal management on architectural or even on OS-level results in a reduction of system's performance. A mixed-integer linear programming formulation to reduce the peak temperature of an application by guaranteeing hard real-time constraints is proposed in [5]. In [14, 24], temperature-aware task allocation algorithms for embedded MPSoCs are explored by comparing various power-aware with thermal-aware heuristics. All these approaches show that both performance and thermal analysis are crucial tasks in the design of multi-processor systems.

Nowadays, the de-facto standard for thermal analysis is low-level simulation. In recent years, both software [4, 9] as well as hardware simulators [1] for MPSoC platforms have been developed that estimate the system's power consumption. Estimating the temperature of a system is typically based on thermal simulation and HotSpot [13] is the most popular member of this group.

The coupling between a low-level power simulation and a thermal simulator results in virtual platforms for evaluating the thermal behavior of a multi-processor system. For example, a platform to test power and thermal management solutions is proposed in [3]. Low-level virtual platforms calculate accurate estimations of the transient temperature evolution of a system, but are limited in terms of execution time. In [10], a hardware/software emulation framework is extended by connecting an FPGA emulation platform with a software thermal analysis library running on a host computer that calculates the current temperature of on-chip components at runtime. Though FPGA emulation platforms tremendously improve the execution time, they require additional hardware, which in turn makes them inflexible and costly. Our approach differs from previous work in two aspects. First, we try to solve it by system-level simulation and thermal calibration, and second, we integrate it in a complete system design flow.

Modern design flows for streaming applications specified as KPN [15] or SDF [18] enable an efficient design of multi-processor systems by providing automatic mapping of applications onto distributed platforms [2]. Besides automated mapping optimization, they enable the reuse of the application for many different platforms. Examples of such multi-processor design flows are Artemis [20], DOL [22], KOSKI [16], and StreamIt [23]. All of them have in common that they support system-level performance analysis, but have a lack in thermal analysis options. Therefore, we integrate the proposed approaches for automated model calibration and system-level thermal evaluation in a complete design flow, that is, DOL.

3. PROBLEM AND APPROACH

In this paper, design space exploration refers to the task of exploring the optimal mapping, that is, binding and scheduling of a multi-processor streaming application onto a distributed memory architecture in a time and thermal efficient manner. By varying the binding of the application elements to computation and communication resources, and the scheduling policies for each computation and communication resource, aimed system properties are optimized, e.g., increasing throughput and decreasing the maximum temperature. Both the system-level specification of the application and the architecture are input to design space exploration, which together with a mapping specification form the synthesizable system specification. Figure 1 illustrates the task of design space exploration as it is considered in DOL [22], and consequently, in this paper.

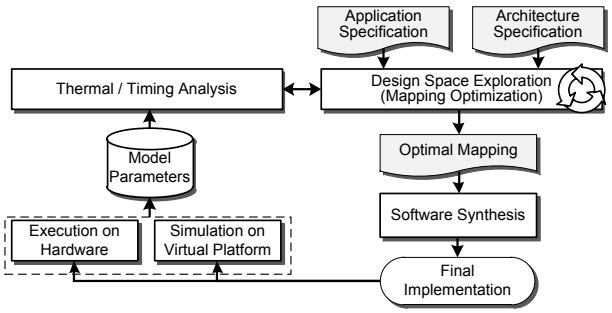

Figure 1: Design space exploration.

The performance and thermal characteristics of every candidate mapping is analyzed with respect to timing and thermal characteristics, respectively, and the output of the design space exploration is a good mapping of a given multi-processor application onto a distributed memory architecture. Even if the task of analyzing timing characteristics of design alternatives has been extensively investigated in recent years (see [2] for a summary), there is a lack of accurate thermal analysis methods at high abstraction levels mainly due to the extensive calibration of such methods.

This paper considers the problem of how to automatically calibrate a system-level thermal analysis model by selecting both timing and thermal parameters that describe the interaction between application and the underlying architecture. The advantage is that the obtained parameters only have to be obtained once for analysis and can be used repeatedly for the evaluation of the candidate mappings during design space exploration. The viability of the proposed approach is demonstrated based on a novel thermal evaluation method that calculates the transient temperature evolution of a candidate mapping by means of calibration data, that is obtained prior to design space exploration. Finally, DOL [22] is extended with the ability to analyze thermal characteristics of candidate mappings targeting the MPARM platform [4].

4. PROBLEM FORMALIZATION

As a general evaluation model does not exist, we restrict ourselves to applications adhering to the synchronous data flow (SDF) [18] model of computation that are executed on a distributed memory architecture. In this section, we formally describe this class of systems as well as the considered problem, to establish a common language.

4.1 System Specification

A multi-processor system is completely specified by the application, architecture, and mapping specification. Figure 2 illustrates the notation by means of an example system.

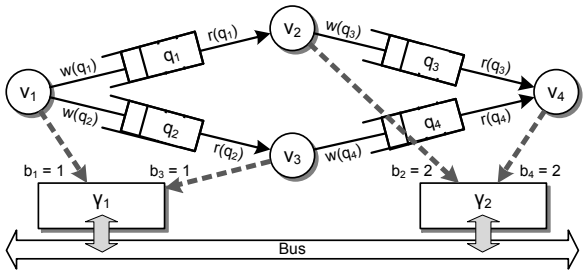

Figure 2: Formal system specification.

Application. The model of computation used in this project is SDF [18], which has the advantage that the computation is separated from communication and therefore, the application behavior is specified independently of the process network [2, 11]. We represent an SDF application as a directed, connected graph $\mathcal{A} = (V, Q, W)$. Every process $v \in V$ represents a node in the graph. Edges are used to model unbounded FIFO channels $q \in Q$, which in turn are used by processes to communicate. $W = \{w_1, \ldots, w_{|Q|}\}$ describes the token consumption behavior. The token consumption w_q of the channel q is represented as a triple (p_e, c_e, d_e). Assuming q connects the nodes v_1 and v_2, the number of tokens produced by node v_1 in every execution is denoted p_e. The second parameter of the triple, c_e is the number of tokens consumed in every execution by v_2 and the initial amount of tokens in the channel q is called d_e and it represents the communication delay.

Architecture. The architecture $\mathcal{T} = \{\gamma_1, \ldots, \gamma_{|\Gamma|}, sc_1, \ldots, sc_{|SC|}\}$ is assumed to be a homogeneous multi-tile system with distributed memory architecture. sc_k denotes a shared component and each tile γ_i is composed of m components $c_{j,i}, j \in \{1, \ldots, m\}$.

Mapping. The mapping of the application onto the architecture consists of the binding b and the scheduling information σ of each processor. The binding can be represented by a vector $b = (b_1; \ldots; b_{|V|}) \in \{1, \ldots, |\Gamma|\}^{|V|}$ where $b_i = k$ if process v_i is assigned to tile t_k.

Power Model. Given an application specification $\mathcal{A}$ and an architecture specification $\mathcal{T}$, the power consumption of component c when process v is executed, is $P_{c,v}$. The power consumption of a tile γ with totally m components is accordingly represented as a vector $\mathbf{P}_{\gamma,v} = (P_{1,v}; \ldots; P_{m,v})$.

Calibration Data. The calibration data is represented by a set $\psi = (\psi_{\text{Time}}, \psi_{\text{Thermal}})$, where ψ_{Time} are timing parameters and ψ_{Thermal} thermal parameters.

4.2 Problem Statement

Using the notation introduced above, the problem of estimating the temperature evolution can be defined as follows:

> *Given a system specification $S = \{\mathcal{A}, \mathcal{T}, (b; \sigma)\}$ consisting of an application specification $\mathcal{A}$, an architecture specification $\mathcal{T}$, and a mapping specification $(b; \sigma)$, generate the transient temperature evolution $\mathbf{T}(t)$ of the system by means of pre-characterized calibration data ψ.*

Similarly, the problem of model calibration can be stated as follows:

> *Given an implementation of the system S, determine the parameters of the calibration data $\psi = (\psi_{Time}, \psi_{Thermal})$.*

5. HIGH-LEVEL SIMULATION

In this section, we answer the question how to represent the thermal behavior of each hardware component when the system specification S, and its timing and thermal parameters ψ are given. For this purpose, we introduce the system-level thermal emulator (SLTE), a novel system-level thermal evaluation model.

5.1 System-Level Thermal Emulator (SLTE)

The proposed system-level thermal evaluation model calculates the transient temperature evolution by individually characterizing subsystems with pre-characterized timing and power values. In particular, in the context of this paper, a subsystem refers to a piece of software that is executed on a specific processor and its behavior is modeled by the average case execution time (ACET) and the average power consumption P_{avg} of the components. By abstracting out the power characteristics of a system as being constant for a specific time interval, the transient power consumption of the system can be calculated much faster than using low-level simulation. Figure 3 provides an overview of SLTE. In addition to the system specification, calibration data ψ is used as input to the temperature evaluation model to characterize the individual subsystems.

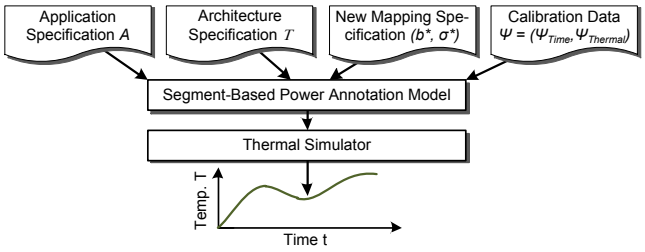

Figure 3: Block diagram of SLTE.

5.2 Segment-Based Power Annotation Model

The core idea of the segment-based power annotation model is to split a process into segments and individually annotate these segments with corresponding power values. When evaluating a new candidate mapping, a two step procedure is executed to estimate the new transient power consumption of the system, see Fig. 4. First, a timetable θ_γ is calculated for each tile $\gamma \in \Gamma$ that describes the order in which the segments are executed subject to the new binding b^* and scheduling σ^*. Afterwards, the transient power consumption is estimated by annotating the timetables with the corresponding power values, that is, $(\theta_\gamma, \psi_{\text{Thermal}}) \mapsto \mathbf{P}_\gamma(t)$.

The level of abstraction used to create the timetables is a trade-off between accuracy and execution speed. Streaming applications have the characteristics that certain components like shared memories are only utilized when a process is reading from or writing to a channel. Therefore, from the thermal perspective it is reasonable to distinguish between *Reading*, *Writing*, and *Computing* segments. In addition to modeling the execution of the processes, the power

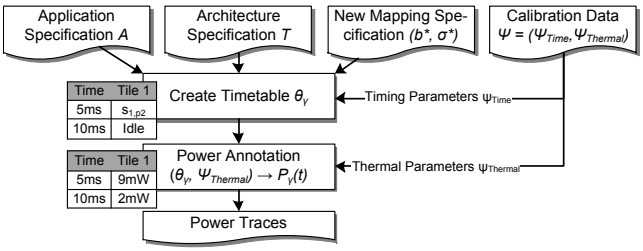

Figure 4: Segment-Based Power Annotation Model

annotation model has to take into consideration the overhead generated by the run-time environment. This overhead mainly arises from two sources: the ability to provide multiprocessing on single processors, and the iterative invocation of processes. Suppose that multi-processing is implemented by kernel- or user-space threads, then the context switch handler consumes CPU resources, as well. This overhead is modeled by multiple segments, that are scheduled whenever a context switch occurs. Finally, the overhead introduced by the iterative invocation of processes is modeled as an additional segment that is scheduled between two iterations.

5.3 Temperature Evaluation

A widely used approach to model the heat transfer of modern VLSI systems is a set of linear first-order differential equations, that results from the duality between heat transfer and an electrical network [17, 21]:

$$\mathbf{C} \cdot \frac{\mathrm{d}\mathbf{T}(t)}{\mathrm{d}t} + \mathbf{G} \cdot \mathbf{T}(t) = \mathbf{P}(t), \qquad \mathbf{T}(0) = \mathbf{T}_0 \tag{1}$$

where $\mathbf{T}$ is the temperature vector, $\mathbf{C}$ the capacitance matrix, $\mathbf{G}$ the conductance matrix, and $\mathbf{P}$ the power vector. Both the capacitance and conductance matrix are assumed to be independent of the temperature.

Sampling the power at a sufficiently small, but constant time interval Δt, the power consumption is constant during the time interval Δt, and the temperature can be discretized by defining $\mathbf{T}[k] = \mathbf{T}(k \cdot \Delta t)$. Consequently, the calculation of the temperature's alteration is reduced to only two matrix-vector multiplications, decreasing the evaluation time compared to typical numerical approaches:

$$\begin{aligned}\mathbf{T}[k+1] &= e^{-\mathbf{A}\cdot\Delta t} \cdot \mathbf{T}[k] + \mathbf{A}^{-1} \cdot \left(\mathbf{I} - e^{-\mathbf{A}\cdot\Delta t}\right) \cdot \mathbf{B} \cdot \mathbf{P}[k] \\ &= \mathbf{E} \cdot \mathbf{T}[k] + \mathbf{F} \cdot \mathbf{P}[k]\end{aligned} \tag{2}$$

where $\mathbf{A} = \mathbf{C}^{-1} \cdot \mathbf{G}$ and $\mathbf{B} = \mathbf{C}^{-1}$.

6. AUTOMATED MODEL CALIBRATION

Nowadays, manual model calibration of system-level thermal simulation models is practically impossible as complex multi-processor systems like video decoders often require thousands of parameters. Therefore, the new challenge is to design methods that automatically parameterize system-level simulation models according to the considered system.

The required parameters of SLTE, that is introduced in the previous section, are summarized in Table 1. In the context of this paper, we categorize the parameters in two subsets, namely timing and thermal parameters. Timing parameters include the execution time and the characteristics of the channels, and thermal parameters include power consumption and thermal configuration of the architecture.

Table 1: Model parameters required by SLTE.

Entity	Parameter	Unit	Source
segment s	average-case execution time ACET(s)	sec / iteration	low-level sim.
	average power consumption P_{avg}	W	low-level sim.
queue q	min. / max. token size $N_{\min}(q), N_{\max}(q)$	bytes / access	functional sim.
	write rate, read rate $w(q), r(q)$	1	functional sim.
processor	clock frequency f	cycles / sec	hardware data-sheet
architecture $\mathcal{T}$	capacitance matrix $\mathbf{C}$	J/K	low-level sim.
	conductivity matrix $\mathbf{G}$	W/K	low-level sim.

6.1 Extracting Timing Parameters

The approaches for obtaining timing parameters are based on the methods presented in [12]. However, instead of extracting all parameters for a process as in [12], the timing parameters are obtained separately for each segment of that process.

Consequently, functional simulation is used to determine time-independent parameters by monitoring the `read` and `write` calls. The time-dependent parameters, namely the average execution time of the segments and the CPU resources consumed by the run-time environment, are obtained by simulating the system in a cycle-accurate simulator.

Additional monitoring is required to obtain the amount of CPU resources consumed by the run-time environment, that is, context switching and process invocation. To illustrate this principle, we discuss the approach to extract the overhead introduced by context switching. Suppose that the process state model can be simplified to three states, namely *active*, *ready*, and *blocked*. Then, the amount of CPU resources required for context switching can be obtained by monitoring the start and end of the methods used for context switching, that is, suspension, interruption, context storing, and context restoring.

6.2 Extracting Thermal Parameters

A two-step procedure is required to extract the power parameters by means of a low-level simulation platform. First, the transient dynamic power consumption of each component is recorded as power traces. Second, the power traces are segmented according to the desired granularity and the average dynamic power consumption of each segment is calculated. Therefore, suppose that segment s is mapped onto tile γ_i that consists of m components, and k components on tile γ_j, $i \neq j$ are involved in communication. Then, the model parameters that describe the power consumption of segment s can be expressed by

$$\begin{aligned}\psi_{\text{Power}} : s \mapsto \{&P_{c_{1,i}}, \ldots, P_{c_{m,i}}, \\ &P_{c_{1,j}}, \ldots, P_{c_{k,j}}, P_{sc_1}, \ldots, P_{sc_o}\}.\end{aligned} \tag{3}$$

As the platform consists of homogeneous tiles, the obtained model parameters for tile γ_i are valid for all other tiles, as

well. Otherwise, the parameters have to be obtained separately for each type of tile.

Finally, the parameters of the architecture's thermal configuration, that is, the capacitance matrix $\mathbf{C}$ and the conductivity matrix $\mathbf{G}$, can be obtained from a low-level thermal analysis model that sets up the thermal differential equation system.

6.3 Automated Parameter Extraction

In order to obtain the parameters for the thermal analysis of a single mapping candidate, a functional simulation is carried out and the system is simulated in a low-level thermal evaluation tool chain. The low-level thermal evaluation tool chain takes as input the system's synthesis specifications, and calculates the execution times, the power consumption, and the thermal configuration of the platform. First, a synthesis tool generates the glue code of the application according to the mapping specification. The application is then executed in a low-level simulator to obtain accurate estimations of the execution times and power consumption of different segments. Finally, the thermal configuration of the architecture is obtained by a low-level thermal analysis model. This approach provides accurate estimations of the transient temperature evaluation, but it is almost infeasible during design space exploration being too expensive in terms of execution time.

Therefore, a viable approach is to collect all parameters before design space exploration. As the 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 architecture. All other parameters obtained by low-level simulation vary for each candidate mapping, but can be estimated by a set of benchmark mappings that covers the entire design space area. As the thermal configuration of the platform is independent of the mapping, both matrices $\mathbf{E}$ and $\mathbf{F}$ can be calculated before design space exploration, as well. Finally, all parameters are stored in a database and during design space exploration, the analysis model just looks up the corresponding parameters.

7. CASE STUDIES

In this section, the viability of the proposed approaches for automated model calibration and system-level thermal evaluation is investigated by considering multi-processor streaming applications that are executed on a virtual platform of multiple ARM processors.

7.1 Experimental Setup

The target platform is the MPARM virtual platform [4], that emulates an ARM-based MPSoC. The reconfigurable platform is composed of a variable number of identical 32-bit ARM 7 processors and shared memories, which are connected by a shared bus. The communication channels are implicitly assumed to be mapped onto the scratchpad of the sender process' tile. In all experiments, thermal management methods have been deactivated, but it is assumed that the core can switch to a power-saving mode if no process is executed.

For the automated model calibration of SLTE with timing and thermal parameters, a prototype implementation of the thermal evaluation tool chain has been implemented based on DOL [22], the MPARM virtual platform, and HotSpot [13]. The considered floor plan and thermal configuration conform to the examples given in [1].

Table 2: Example applications.

Application	# tiles	# processes	# segments
Matrix multiplication	3	10	152
IIR filter	2	3	18
FFT	3	14	284
MJPEG impl. 1	3	5	9762
MJPEG impl. 2	4	8	832

7.2 Evaluation Results

Accuracy and Speedup. First, we discuss the accuracy and speedup of SLTE by simulating four multi-processor streaming applications: a distributed matrix multiplication, infinite impulse response (IIR), fast fourier transformation (FFT), and motion JPEG (MJPEG), see Table 2 for an overview of their complexity. The speedup is measured as ratio between the execution time of the system when simulating it with the low-level simulation tool-chain and with our SLTE. We quantify the accuracy of SLTE by means of the maximum temperature of the system. Therefore, we introduce the pessimism, that measures the normalized distance between the thermal low-level simulation result and the result obtained with SLTE:

$$pessimism = \frac{|\max(T_{\mathrm{SLTE}}) - \max(T_{\mathrm{sim}})|}{margin} \qquad (4)$$

where *margin* is the thermal difference between the maximum temperature when executing the system with utilization 100% and 0%, $\max(T_{\mathrm{SLTE}})$ is the maximum observed temperature in the SLTE simulation and $\max(T_{\mathrm{sim}})$ the maximum observed temperature in the low-level simulation. The pessimism is calculated separately for each component and an average of all pessimisms is used as metric for the accuracy.

All example applications are simulated for about three seconds and the invocation interval of the source process is selected so that all processes can complete their current iteration before the next one starts. Figure 5 shows the measured speedups and accuracies of the four example applications each for various candidate mappings. Every example application is calibrated with a selected benchmark mapping and the obtained calibration data is later used for the evaluation of different candidate mappings in SLTE. The average speedup for all experiments is 1463 and the corresponding average pessimism is only 0.57 %. This justifies our statement that accurate results about the thermal characteristics of multi-processor systems can be obtained from system-level thermal simulation models provided that detailed calibration data are available. The two implementations of the MJPEG application show the limitations of the proposed system-level thermal evaluation method. Fine-grained applications often consist of a huge amount of segments, and consequently, the setup of the analysis model and the scheduling creation becomes time-consuming, which in turn reduces the speedup.

Design Space Exploration. In the second case study, an example of use for the proposed system-level thermal evaluation method is studied. When optimizing the throughput of a multi-processor streaming application on a given distributed platform subject to the maximum temperature,

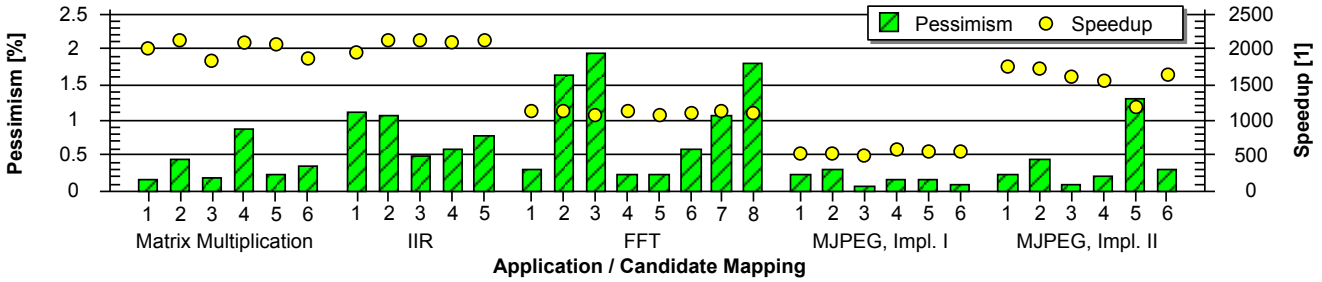

Figure 5: Pessimism and speedup of various multi-processor streaming applications.

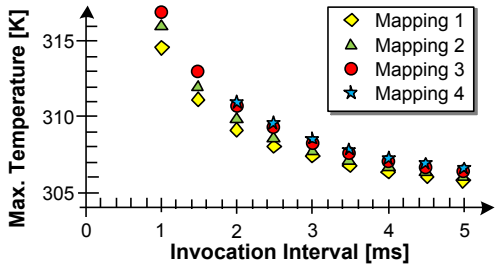

Figure 6: Design space of FFT application.

the only degrees of freedom are the mapping and the invocation interval. In Fig. 6, the design space of a distributed implementation of the FFT application has been explored. Only invocation intervals in which all processes can complete before its next iteration starts, are plotted and the maximum temperature of a candidate mapping is the highest temperature that is ever observed on all components of the system. As no thermal management is applied, the difference between the maximum temperatures of two candidate mappings with the same invocation interval is relatively small. Even though only presenting the results for the FFT application, similar results are achieved with the other example applications. The graph shows the trade-off between throughput and maximum temperature and enables the designer to exclude all design alternatives that overshoot the critical temperature of the architecture.

8. CONCLUSIONS

In this paper, the automated model calibration of a system-level thermal analysis models has been considered. The proposed method uses functional, low-level, and thermal simulation to obtain the characteristical timing and thermal parameters of a multi-processor system that is only given by its system synthesis specifications. In order to illustrate the advantages of automated model calibration, a novel system-level thermal analysis method is proposed that models the power consumption segment-based. Various case studies show that the proposed system-level thermal evaluation method reduces the evaluation time on average by more than three magnitudes. Together with automated model calibration, this method enables the integration of thermal analysis into software synthesis for multi-processor systems.

Acknowledgments

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

References

[1] D. Atienza et al. HW-SW Emulation Framework for Temperature-Aware Design in MPSoCs. *ACM T. Design Automation of Electronic Systems*, 12(3):1–26, 2007.

[2] I. Bacivarov et al. Methods and Tools for Mapping Process Networks onto Multi-Processor Systems-On-Chip. In *Handbook of Signal Processing Systems*. Springer, 2010.

[3] A. Bartolini et al. A Virtual Platform Environment for Exploring Power, Thermal and Reliability Management Control Strategies in High-Performance Multicores. In *Proc. GLSVLSI*, 2010.

[4] L. Benini et al. MPARM: Exploring the Multi-Processor SoC Design Space with SystemC. *J. VLSI Signal. Proces.*, 41(2):169–182, 2005.

[5] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs. In *Proc. DATE*, 2008.

[6] A. K. Coskun et al. Dynamic Thermal Management in 3D Multicore Architectures. In *Proc. DATE*, 2009.

[7] A. K. Coskun, T. S. Rosing, and K. Whisnant. Temperature Aware Task Scheduling in MPSoCs. In *Proc. DATE*, 2007.

[8] J. Donald and M. Martonosi. Techniques for Multicore Thermal Management: Classification and New Exploration. In *Proc. ISCA*, 2006.

[9] N. Eisley, V. Soteriou, and L. Peh. High-Level Power Analysis for Multi-Core Chips. In *Proc. CASES*, 2006.

[10] P. Garcia del Valle and D. Atienza. Emulation-Based Transient Thermal Modeling of 2D/3D Systems-on-Chip with Active Cooling. *Microelectronics J.*, 41(10):1–9, 2010.

[11] D. Gelernter and N. Carriero. Coordination Languages and Their Significance. *Commun. ACM*, 35:97–107, 1992.

[12] W. Haid et al. Generation and Calibration of Compositional Performance Analysis Models for Multi-Processor Systems. In *Proc. SAMOS*, 2009.

[13] W. Huang et al. HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design. *IEEE T. VLSI Sys.*, 14(5):501–513, 2006.

[14] W.-L. Hung et al. Thermal-Aware Task Allocation and Scheduling for Embedded Systems. In *Proc. DATE*, volume 2, 2005.

[15] G. Kahn. The Semantics of a Simple Language for Parallel Programming. In *Proc. of the IFIP Congress*, 1974.

[16] T. Kangas et al. UML-Based Multiprocessor SoC Design Framework. *ACM T. Embed. Comput. Syst.*, 5:281–320, 2006.

[17] A. Krum. Thermal Management. In F. Kreith, editor, *The CRC Handbook of Thermal Engineering*. CRC Press, 2000.

[18] E. Lee and D. Messerschmitt. Synchronous Data Flow. *Proc. IEEE*, 75(9):1235–1245, 1987.

[19] S. Murali et al. Temperature-Aware Processor Frequency Assignment for MPSoCs Using Convex Optimization. In *Proc. CODES+ISSS*, 2007.

[20] A. Pimentel. The Artemis Workbench for System-Level Performance Evaluation of Embedded Systems. *Int'l J. Embedded Systems*, 3(3):181–196, 2008.

[21] K. Skadron et al. Temperature-Aware Microarchitecture. In *Proc. ISCA*, 2003.

[22] L. Thiele et al. Mapping Applications to Tiled Multiprocessor Embedded Systems. In *Proc. ACSD*, pages 29–40, 2007.

[23] W. Thies, M. Karczmarek, and S. Amarasinghe. StreamIt: A Language for Streaming Applications. In *Compiler Construction*. Springer, 2002.

[24] Y. Xie and W.-l. Hung. Temperature-Aware Task Allocation and Scheduling for Embedded Multiprocessor Systems-on-Chip (MPSoC) Design. *J. VLSI Signal. Proces.*, 45(3):177–189, 2006.

[25] J. Yang et al. Dynamic Thermal Management Through Task Scheduling. In *Proc. ISPASS*, 2008.