Online Convex Optimization-Based Algorithm for Thermal Management of MPSoCs

Francesco Zanini
Integrated Systems
Laboratory (ISL)
EPFL
Lausanne, Switzerland
francesco.zanini@epfl.ch

David Atienza
Embedded Systems
Laboratory (ESL)
EPFL
Lausanne, Switzerland
david.atienza@epfl.ch

Giovanni De Micheli
Integrated Systems
Laboratory (ISL)
EPFL
Lausanne, Switzerland
giovanni.demicheli@epfl.ch

Stephen P. Boyd
Information Systems
Laboratory (ISL)
Stanford University
Palo Alto, CA, U.S.A.
boyd@stanford.edu

ABSTRACT

Meeting the temperature constraints and reducing the hot-spots are critical for achieving reliable and efficient operation of complex multi-core systems. The goal of thermal management is to meet maximum operating temperature constraints, while tracking time-varying performance requirements. Current approaches avoid thermal violations by forcing abrupt operating points changes, which cause sharp performance degradation. In this paper we aim at achieving an online smooth thermal control action, that minimizes the tracking error. We formulate this problem as a discrete-time optimal control problem, which can be solved via online by using an embedded convex optimization solver using a receding horizon approach. The optimization problem considers the thermal profile of the system, its evolution over time, current and past time-varying workload requirements. We perform experiments on a model of the 8-core Niagara-1 multicore architecture, which show that the proposed method outperforms state-of-the-art thermal management approaches by enabling performance speed-ups of up to 2.5× and improvements up to 12× and 3.4× in relation to frequency and temperature variations over time, respectively.

Categories and Subject Descriptors

B.4.m [**Hardware**]: Performance and Reliability—*Miscellaneous*

General Terms

Management, Algorithms

Keywords

Thermal Management Online Convex Optimization MPSOCs

1. INTRODUCTION

Thermal management techniques are receiving a lot of attention. Many state-of-the-art thermal control policies manage power consumption via *dynamic frequency and voltage scaling* (DVFS) [3].

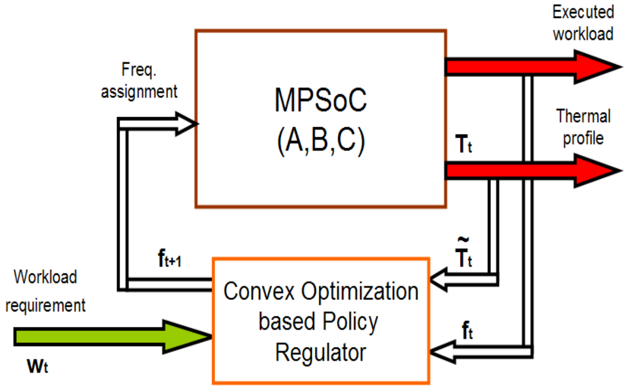

Figure 1: Diagram of proposed Convex Optimization-based thermal management policy

DVFS can be targeted to power density reduction, which has the effect of reducing overall temperature. However, these techniques do not directly minimize thermal gradients or hot-spots [2], [7].

Temperature gradients are a concern not only in space, but also in time. The frequent abrupt change in working frequencies and voltages produces thermal cycling that raises the failure rate of the system [10]. The effect of thermal cycling on the reliability of a chip can be modelled by the Coffin-Manson relation, which relates in an exponential way the number of cycles to failure to the magnitude of thermal cycling [11]. In addition, abrupt power-mode transitions, waste additional power [12].

The proposed approach addresses the frequency assignment of an MPSoC as a convex optimization problem. The optimization is a joint optimization that takes into account the heat propagation model of the MPSoC, the reliability information related with the workload prediction method and finally future consequences of present actions over L future time steps. The proposed system dynamically adapts to the actual run-time situation of the system, without relying on any exhaustive characterization at design time of the possible workloads of the target system. The optimal controller predicts the future thermal trajectory of the system, and maximizes performance by completely satisfying thermal constraints. The optimization process requires a small computational complexity and so it can be performed online. The result of this optimization is a very smooth control where both satisfied thermal reliability constraints while achieving significant performance improvements compared with state-of-the-art methods.

GLSVLSI'10, May 16–18, 2010, Providence, Rhode Island, USA.
Copyright 2010 ACM 978-1-4503-0012-4/10/06 ...$10.00.

2. PREVIOUS WORK

Many researchers in computer architecture have recently focused on thermal control for *Multi-Processor System on Chips* (MPSoCs) [7], [3]. Processor power optimization and balancing using DVFS have been proposed in several works [3]. In [7] a significant reduction in localized hotspots has been obtained using thread migration techniques. In [2] a thermal/power model for super-scalar architectures is presented.

The authors of [5] propose a convex optimization based approach. The input parameters needed for the optimization are the thermal profile, chip physical parameters and scheduler requirements. However, apart from chip parameters, the other two input data can assume many values. Assumptions needed to make the system feasible from an implementation perspective degrade the quality of results.

In [6] the policy is formulated as a discrete-time optimal control problem, which can be solved using the theory and computational tools developed in the field of model-predictive control. The policy is computed on-line by multiplying the vector containing current thermal profile information and workload requirements by pre-computed coefficients. The main problem with this approach is that the number of coefficients to store is usually large for a complex MPSoC system. As a result, this method can be implemented only in MPSoCs described by thermal models using a small number of cells (in the order of tenths) while the proposed method can manage MPSoC modelled with a higher number of cells. In addition to that the policy does not take into account the past history of the task arrival process to predict future workload requirements.

In two recent works, the idea of exploiting history information has been exploited to improve thermal management policies. In [14] the policy exploits a temperature forecast technique base on a auto-regressive moving average model. In [15] the authors propose a novel technique that adapts the thermal management policy to the current workload characteristics. The adaptation is done online exploiting information related to the workload history. The problem with both these techniques is the following. They do not completely avoid hot spots, but they simply reduce their frequency. The reason is that the interaction between the prediction method, the thermal behavior of the MPSoC and the frequency assignment of the MPSoC has not been addressed as a joint optimization problem. The action taken by the policy to avoid hot spots does not address the problem from a global optimum perspective.

3. DESIGN METHODOLOGY

3.1 High-Level System Description

The block diagram of the proposed control system is shown in Figure 1. The regulator monitors the MPSoC state composed by temperature values and working frequencies. The temperature state at time t is defined as a vector $T_t \epsilon \Re^{2n}$, where $(T_t)_i$ is the temperature of cell i at time t. The thermal model is composed by two layers, each one composed by n cells. For this reason the total number of cells representing the MPSoC thermal model is $2n$. The frequency state at time t is defined as a vector $f_t \epsilon \Re^p$, where $(f_t)_i$ is the frequency value of input i at time t and p is the number of inputs. Working frequencies are controlled by the regulator, and are known while temperatures are monitored by on-die thermal sensors. Temperature measurements at time t are defined as a vector $\tilde{T}_t \epsilon \Re^s$, where $(\tilde{T}_t)_i$ is the temperature measurement coming from sensor i at time t. s is the number of thermal sensors inside the MPSoC. Thus, the current state of the system T_t at time t is generated from data derived from real thermal sensor measurements $\tilde{T}$ on the real MPSoC. This approach has the advantage of avoiding the propagation of some inaccuracies between the model and the real system.

The regulator receives a workload requirement from higher-level software layers (e.g., operating system or OS). At time t, the workload is defined as a vector $w_t \epsilon \Re^p$, where $(w_t)_i$ is the frequency that input i at time t should have in order to satisfy the desired performance requirement coming from the scheduler. The regulator provides a frequency assignment that minimizes the *tracking error*. This error is proportional to the difference between the offered and required workload. In fact, *the tracking error is a direct measure of performance penalty*, as it is greater than zero when the controller sets processor working frequencies not exactly matching the requests coming from the OS.

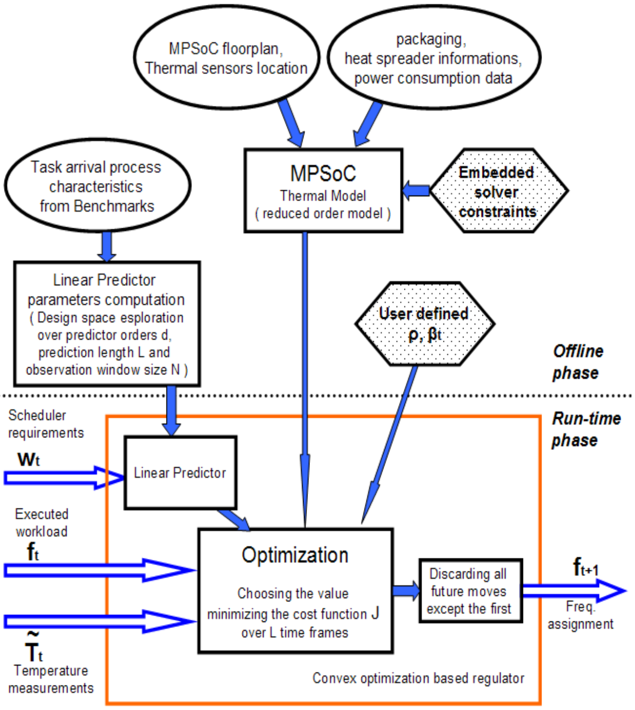

Figure 2: Design flow of the proposed Convex Optimization-based thermal management policy

Constraints on the maximum temperature of the MPSoC need to be also enforced in the optimization process. Then, the optimal control problem is formulated over an interval of L time steps, which starts at current time t. For this reason, our approach is predictive [4]. The workload requirement in the future L time steps is predicted using a linear model described in detail in following section. The result of the optimization is an optimal sequence of future control moves (i.e., frequency settings for the cores). Only the first sample of such a sequence is actually applied to the process; the remaining moves are discarded. At the next time step, a new optimal control problem based on new temperature measurements and required frequencies is solved over a shifted prediction horizon. Such a *receding-horizon* mechanism represents a way of transforming an open-loop design methodology (i.e., the convex based policy proposed in [5]) into a feedback one, as at every time step the input applied to the process depends on the most recent measurements.

The way the problem has been formalized using convex models enables a fast and easy numerical computation [16]-[17]. The experimental results show significant advantages of the proposed method in terms of both performance and reliability over state-of-the-art methods (see Section 4.4).

3.2 Proposed System Operation

The procedure is summarized in the following block diagram of Figure 2. The operation of the system can be divided into two phases: an off-line design phase and run-time one.

The first and off-line phase is described by the block diagram in the upper section of Figure 2. During this phase the convex optimization based regulator is designed. The floorplan of the MPSoC, including thermal sensor locations, are obtained as inputs. Packaging information including heat spreader properties and power consumption correlation data are included as well in the procedure. The time period at which the policy needs to be applied is also obtained as an input. All previously mentioned data are used to create the MPSoC thermal model. This model is then simplified using a Hankel singular value-based model order reduction (see Section 3.3.2 for more details). The reduction process is performed according to performance constraints defined by the designer to include an embedded solver able to compute the solution to the convex optimization problem in real-time. Even though this algorithm usually takes few microseconds [16, 17] on normal PC cores, models with a large number of states can have a prohibitive cost for embedded systems where computational capabilities are limited. Thus, during this off-line phase, the parameters of the linear predictor are obtained as well by testing the predictor using execution characteristics derived from benchmarks. Finally, user-defined parameters used in the optimization process are specified during this phase.

The second phase is called run-time phase. This phase is described by the block diagram in the lower section of Figure 2. This phase shows the operation of the convex optimization based regulator during its run-time execution. Input data coming from the scheduler are used by the linear predictor to generate a prediction on future requests. The prediction is made up to L steps in the future and considering the last N samples. The predictor is made using a polynomial least squares estimator of order d. All these parameters have been derived during the off-line design phase of the regulator. Reliability information on the just generated prediction as well as the prediction itself are sent in real time to the convex solver. Information of temperature measurement coming from thermal sensors and the current frequencies setting of the MPSoC are sent as well. At this stage the convex solver finds the optimum frequency assignment for the inputs of the MPSoC system that will maximize performance under temperature constraints. This is accomplished by minimizing a cost function J under some specific constraints. This is described with more details in next section. This operations, according to current technology takes few tenth of microseconds [16]. This time is from 3 to 4 orders of magnitude smaller compared with the time the policy is applied (10ms-100ms). The result of the optimization is an optimal sequence of future control moves (i.e., frequency settings for the cores). Only the first sample of such a sequence is actually applied to the process, the remaining moves are discarded. This mechanism represents a way of updating the optimization with the most recent thermal measurements.

3.3 Problem Formulation

3.3.1 Frequency Input Model

The frequency input at time t is defined as a vector $f_t \in \Re^p$, where $(f_t)_i$ is the frequency value of input i at time t and p is the number of inputs. Working frequencies represent our optimization variable. Working frequencies are assumed in our model to be continuous and ranging from zero to a maximum frequency value $f_{\max}$ as described formally by Equation 1.

$$0 \preceq f_t \preceq f_{\max} \;\forall\, t \tag{1}$$

where the symbol $\preceq$ means element-wise comparison. The frequency vector represents our optimization variable. The value of its element is assigned by solving the minimization problem described in following subsections. This minimization will try to achieve the desired performance requirements while completely satisfying thermal constraints.

At time t, the relation between the power dissipation $p_t \in \Re^p$ and the frequency of operation f_t is expressed by Equation 2.

$$f_t^{\alpha} = p_t \;\forall\, t \tag{2}$$

where the constant α is chosen depending on the technology and usually it varies from 1 to 2. If $\alpha = 1$, we have a linear dependence (i.e., frequency scaling) while if $1 < \alpha \leq 2$ we obtain a quadratic or sub-quadratic dependence (i.e., DVFS) [5]. In this work we implement both voltage and frequency scaling, for this reason we assume $\alpha = 2$.

3.3.2 Heat Propagation Model

Our thermal model is based on finite-difference analysis as commonly used by well-known system-level thermal analysis tools [2, 8]. Two layers have been used on the vertical direction, namely, the silicon layer and the copper layer. The copper layer here models both the metallization layers and the heat spreading copper layer. Then, the chip floorplan has been divided into several thermal cells of cubic shape, and every single functional unit in the floorplan can be represented by one or more thermal cells of the silicon layer. Thermal modelling is computed considering heat conductances and capacitances of the cells, as computed and validated in [2] and [8].

Any linear system can be represented in state-space form. In our case, by measuring all temperatures having as reference the room temperature, the heat propagation process can be represented in the following way:

$$T_{t+1} = AT_t + Bp_t \tag{3}$$

$$\tilde{T}_t \simeq CT_t \tag{4}$$

At time t, the temperature of the next simulation step of cell i $(T_{t+1})_i$ can be computed thanks to Equation 3. The relation between the frequency assignment at time t and the power consumption is expressed by Equation 2. Matrices $A \in \Re^{2n\times 2n}$ and $B \in \Re^{2n\times p}$ are the ones describing heat propagation properties of the MPSoC. Equation 4 describes the location of temperature sensors inside the MPSoC. Matrix $C \in \Re^{s\times 2n}$ relates the temperature value of each cell with the temperature measurement of a particular sensor. The non-idealites of thermal sensors are taken into account int the formulation by indicating the $\simeq$ symbol in the relation of Equation 4.

The way the model has been described before requires a state for every block composing the floorplan. This requirement is expensive in terms of computational requirements for high accuracy MPSoC models. To reduce the number of the states, we first performed a model reduction using a Gramian-based balancing of state-space realizations [13]. After that, we reduced the order of the state-space model by eliminating the states with corresponding small Hankel singular values. The full MPSoC model is now described by the following system of equations:

$$\tilde{X}_{t+1} = \tilde{A}\tilde{X}_t + \tilde{B}p_t \tag{5}$$

$$\tilde{T}_t \simeq \tilde{C}\tilde{X}_t \tag{6}$$

where l is the number of states of the new reduced order model, matrix $\tilde{A} \in \Re^{l\times l}$ and matrix $\tilde{B} \in \Re^{l\times p}$. Equation 5 describes the state update for the reduced order model of the MPSoC. This equation is analogous to Equation 3. The only difference is that, in this case, the states do not represent directly temperature values inside each cell.

Matrix $\tilde{C} \in \Re^{s\times l}$. Equation 6 relates the value of the states with temperature measurements in s specific locations inside the MPSoC. This equation is analogous to Equation 4 and describes how the value of states in the reduced system can be derived from real temperature measurements in limited specific locations inside the MPSoC. To be able to perform this last operation, matrix $\tilde{C}$ should be with a number of rows s greater or equal to the number of columns l and needs to have a rank, at least, equal to l.

3.3.3 Workload Model

The workload requirement at time t is defined as a vector $w_t \in \Re^p$, where $(w_t)_i$ is the workload requirement value for input i at time t. $(w_t)_i$ is the frequency that input i at time t should have in order to satisfy the desired performance requirement coming from the scheduler. In our model it is assumed to be continuous and ranging from zero to a max value $f_{\max}$ as described formally by Equation 7.

$$0 \preceq w_t \preceq f_{\max} \;\forall\, t \tag{7}$$

where the symbol $\preceq$ means element-wise comparison.

The way we measure the performance of the system in achieving the requested workload requirements at time t is given by the vector $u_t \in \Re^p$, namely

$$u_t = w_t - f_t \tag{8}$$

We call u_t undone workload and it expresses the difference between the requested workload and the workload that is actually executed by the MPSoC.

3.3.4 Minimization Objective Function

The issue we have to address is the performance optimization and power minimization problem of a linear time-discrete system subjected to constraints. Constraints are the ones described in previous subsections and they are related to performance requirements to be satisfied and hotspot prevention. The optimization minimizes the power consumption as well as the difference between the workload required by the scheduler and the one actually executed by the cores. This is done by having a maximum temperature constraint on the overall MPSoC. Because of the quadratic relation between the power consumption and the working frequency, the proposed system, to minimize power, will also try to average the power consumption in time among all the cores. This will lead to avoid not uniform thermal profile scenarios during the runtime execution of the system. The frequency assignment problem here is solved by minimizing function J presented below:

$$\begin{aligned} minimize\ J &= \sum_{t=1}^{L} \gamma_t u_t + \rho \sum_{t=1}^{L} 1^T p_t && (9) \\ subject\ to &: \ 0 \preceq f_t \preceq f_{\max}\ \forall\, t && (10) \\ & \tilde{X}_{t+1} = \tilde{A}\tilde{X}_t + \tilde{B}p_t\ \forall\, t && (11) \\ & \tilde{C}\tilde{X}_t \preceq T_{\max}\ \forall\, t && (12) \\ & u_t \succeq 0\ \forall\, t && (13) \\ & u_t = w_t - f_t\ \forall\, t && (14) \\ & p_t \succeq f_t^{\alpha}\ \forall\, t && (15) \end{aligned}$$

Function J is expressed by two sums where the summation index t ranges from 1 to L. During these L future steps the system tries to minimize the cost function J and compute the frequency assignment for these steps.

The first term $\sum_{t=1}^{L} \gamma_t u_t$ is a weighted sum of the amount of predicted required workload that has not been executed. The weight function $\gamma_t \epsilon \Re^p$, with $1 \leq t \leq L$. The way this function is computed and the workload prediction has been estimated will be discussed in detail in next section. Since the system cannot execute jobs that have not arrived, every entry of u_t has to be greater than or equal to 0 as stated by Equation 13, where the symbols $\succeq$ and $\preceq$ stand for element-wise inequality relations. Equation 12 states that temperature constraints should be respected at any time and in any specified location. The second term $\rho \sum_{t=1}^{L} 1^T p_t$ is a weighted sum of the amount of power ($1^T p_t$) required by the system at time t to execute the predicted workload requirement from $t = 1$ to $t = L$. Where ρ is a constant that quantifies the importance that power minimization has in the optimization process.

Every time the minimization is performed, a sequence of optimal control actions is provided for the next L steps. However, only the first step is computed and all the other moves are discarded. Such a mechanism represents a way of creating a feedback loops where the solution is updated with the most recent measurements. This formulation performs a frequency assignment by embedding not only a prediction on the chip temperature profile but also on the future workload to be executed. This prediction and its reliability information are embedded into the optimization process.

The power equation 2, using $\alpha = 2$, is a non-convex equality. Because of the fact that all constraints in the minimization problem of Equation 9 must be convex functions, we relaxed equation 2 to the convex inequality of Equation 15. By doing this operation we changed the original minimization problem to the problem described by the convex Equations 9-15. It can be shown that the resulting relaxed convex problem is equivalent to the original problem with the equality constraint [18]. The reason is because in the optimization problem, we are minimizing a function J that includes p_t in a summation. This means that by minimizing J, we are minimizing p_t as well, and the inequality of Equation 15 will always converge to an equality and the two problems are equivalent [18]. In the proposed problem formulation all equations are convex models and they can be solved with polynomial (in the number of variables and constraints) time complexity using interior point methods [18]. Equation 10 allows a continuous range of frequency settings but this does not prevent from adding in the optimization problem a limitation on the number of allowed frequency values.

To solve the models, we use CVX [19], an efficient convex optimization solver. For our experimental set-up, the Matlab version of the solver takes around 1 second to determine the optimal solution. C++ optimized implementations of this software take few microseconds to run on a state-of-the-art solver [16]-[17], which is at least 3 orders of magnitude less than the interval between two consequent application of the policy (from 10ms to 100ms). In addition, it has been also shown that in much more complex systems that the one we are considering in our case study, with up to 100 optimization variables, the solver takes less than $0.5ms$ on a normal PC platform [17].

3.4 Workload Prediction and its Reliability

To increase the performance of our proposed policy, history information about the task arrival process are exploited by the proposed algorithm. These data are used to make prediction on future workload requirements.

The fact of including these information in the optimization process represents the key advantage of the proposed method versus previous state-of-the-art policies. These inclusions in the problem formulation play a key role in the the optimization process. The policy can indeed forecast future trends of the workload and so the design space of the controller increases. This will increase the options the controller has to achieve its goals. The fact of having such a large design space with many parameters and variables makes this approach unfeasible for explicit solvers such as the one presented in [6]. However in this work we solved the problem by using an embedded solver.

The prediction is done by using a linear model to perform a best d^{th} order polynomial fit. The reason for using a linear model is because, usually, the prediction length L, for this application is short and ranges usually from 1 to 9 samples. The polynomial fit is performed by minimizing the error within the observed window of temperatures, by using the following function:

$$\|w - \check{A}x\|_2^2 \qquad (16)$$

where w_t contains the frequency requirements $\forall t = 1...N$, where N is the length of the observation window of historical data. Matrix $x_t \epsilon \Re^{d+1}$ and vector $\check{A} \epsilon \Re^{d+1}$ are used in the polynomial interpolation process. Equation 16 can be solved as a least squares minimization problem to derive vector $\check{A}$. The prediction on the future workload requirement is performed by assuming that the linear model just derived will hold for the next L data samples. Assuming this assumption hold, the future workload requirement is given by following equation:

$$w_t = \check{A}x_t, \quad \forall \ \ N \leq t \leq N + L \qquad (17)$$

where $w_t \epsilon \Re^p$ for $t > N$ is the predicted workload requirement at time t. We tested the predictor on the benchmarks described in the experimental setup section and we achieved good accuracies for short-term forecasts (L ranging from 1 to 9).

The way we take into account the accuracy of this prediction is embedded in the way the weighting vector γ_t is computed. It is defined according to Equation 18:

$$\gamma_t = \beta_t - \|w_t - \hat{w_t}\| \ \forall \ N - L \leq t \leq N \qquad (18)$$

where $\hat{w}_t$ is the workload predicted by the aforementioned linear predictor and w_t is the actual value of it. The absolute value of difference between the two represents the prediction error. $\beta_t \epsilon \Re^p$ is a vector that adds a different penalty for the workload that has been predicted, but not executed yet, in a different and future time frame. This penalty function can be chosen to be linear, quadratic, exponential or in any other way, according to the impact that a delayed execution of tasks has on performance. In fact, the more reliable the prediction is, the smaller the prediction error is and so the bigger γ_t is. This means that, since in our formulation the prediction is reliable, importance is given to the cost function corresponding to that future time frame. Values of N and d providing the best prediction depend on the workload requirement (task arrival process) statistical properties. These kind of processes are usually non-stationary and depend on the interaction between the user and the MPSoC itself. For the aforementioned reasons, we have chosen these parameters to achieve a good prediction, according to empirical studies performed on different benchmarks [14] for representative examples of the MPSoCs under study in this work.

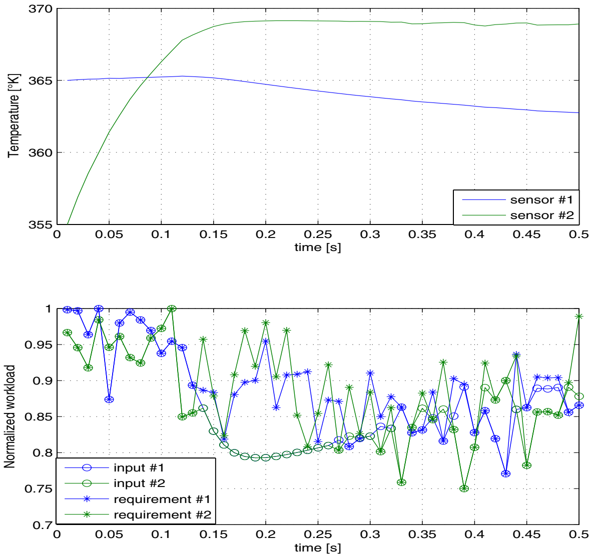

Figure 3: Run-time execution behavior of the prop. method

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

In our experimental setup, we consider an architecture resembling the 8-core Niagara-1 (UltraSparc T1) architecture from Sun Microsystems [1]. This architecture has a maximum operating frequency of 1.2 GHz and the maximum power consumption of each processor core at this frequency to be 4 W [1]. To implement the voltage and frequency scaling techniques, we use working frequencies from 0 to 1.2GHz. We used $\alpha = 2$ [5]. The floorplan of the Niagara-1 multicore architecture, is presented in [6]. The floorplan has been modelled using blocks of 3mm side each, and values of technological parameters and coefficients have been derived from [9], and [1]. There is a large variation in switching activities characteristics of different benchmarks. For this reason, to simulate the system we use the execution characteristics of tasks from a mix of different benchmarks, ranging from web-accessing to playing multimedia [14], [15]. We verified our simulator using the Hotspot simulator [2].

In all our experiments, the proposed convex-based thermal management policy is applied every $10ms$, while the simulation step for the discrete time integration of the RC thermal model has been set to 200μs. The MPC policy has been targeted to track the required workload signal, minimizing power consumption while respecting the maximum temperature limit set to $370^\circ K$. The linear predictor has been designed using a 3^{rd} order polynomial equation, an observation window of $600ms$ and a prediction length equal to $50ms$ in the future. The constant Q has been set to 1 (i.e., $10ms$) and the matrix D has been chosen according to a linear range of weight ranging from 1 to 5 (i.e., $10ms$ to $50ms$).

We assumed to have two frequency inputs controlling the MPSoC. The first frequency input controls cores 1, 4, 5 and 8. The second input sets the frequency value for cores 2, 3, 6 and 7. All the other functional units consume a power that is related to the average power consumption of the cores [1]. We suppose that the scheduler tries to perform a workload balancing strategy on the cores in order to have all the cores running with potentially the same (or very similar) active frequency. By doing this we can assume that in this case the power consumption of the cores depends mostly on their frequency setting. Because of this fact, since in our case study we run half of them with one frequency and the other half with another one, given the symmetry of the structure, we can use only 2 thermal sensors placed on core 1 and core 2 to determine the hottest points in the temperature profile of the MPSoC.

4.2 Run-time Execution Behavior

In the first set of experiments we explore how our policy controls temperature on the Niagara case study in the case of a potential hotspot scenario. Simulation results are reported in Figure 3, where the time domain plot of the normalized requested workload and the temperature measurements for each of the two sensors are reported. The normalized workload is proportional to the frequency setting of the MPSoC; thus, a variations in its value corresponds a variation in the voltage and frequency setting of the cores.

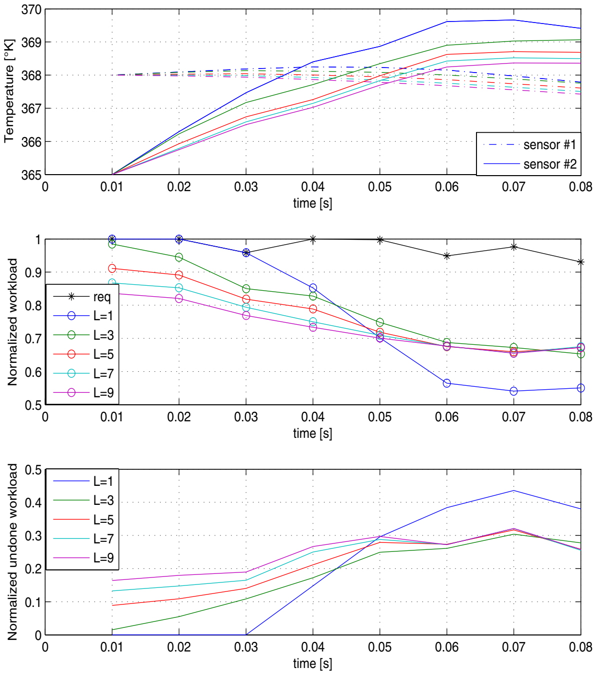

Figure 4: Behavior of the prop. policy for different horizons

In Figure 3, the scheduler requires different workloads for any of two inputs in a very unbalanced way. Temperature measurements for sensor 1 and sensor 2 are reported in the upper plot of Figure 3. In the graph of Figure 3, it can be noted how at $0.15s$ the controller is not able any more to satisfy the requested workload without breaking temperature constraints. For this reason it basically stops following the dynamics of the requested workload but applies a smooth control law that never makes the maximum chip temperature exceed the threshold set to $370^\circ K$. To achieve this, in case of potential overheating, the regulator decreases the frequency of the hottest cores in a way that dynamically adapts the system to achieve maximum performance while respecting temperature constraints. Differently from most state-of-the-art methods, the proposed algorithm uses prediction techniques based not only on the system dynamics but also on past workload requirements history. This can be noted at $0.14s$ where the proposed system decreases the frequency of the cores even if the temperature constraint would have been satisfied by having the requested workload request fulfilled.

4.3 Influence Of The Prediction Horizon

The top graph plots temperature measurements for the two sensors. The second graph plots the overall workload requested to the MPSoC and the workload offered by the system for different prediction horizons L ranging from 1 to 9. Workload measurements in the plot have been normalized to 1. The bottom graph plots the difference between the requested workload and the one provided by the system for the horizons analyzed in the central graph.

By looking at the top graph, in all the analyzed scenarios, the maximum temperature of the MPSoC never exceeds the threshold set to $370^\circ K$. The control policy we are proposing is able to detect this scenario at least $30ms$ in advance and also avoids it by limiting the performance loss due to this constraint. It can be noted also that the higher the prediction horizon, the lower the maximum recorded temperature is. The highest temperature profile is recorded in the case where the prediction horizon has been set to 1 (blue line). By looking at the bottom graph, up to $0.03s$ the curve corresponding to the shorter prediction horizon ($L = 1$) is able to track perfectly the requested workload, however, from 0.05 to the end of the simulation, it shows the worst performance by having the biggest gap between the requested workload and the one provided by the MPSoC. On the some graph, the curve with $L = 9$, even if it does not provide a complete fulfillment of the requirements during the overall simulation, it is the one that provides the smallest undone workload in almost all simulation steps. By analyzing the top and the central graph of Figure 4, it can be noted also that the longer the horizon, the smoother the control action is, and the smoother the temperature profile is.

The reason of previously mentioned behaviors is that the shorter the prediction horizon is, the less the policy can forecast future trends of the workload and so the less powerful the technique is. The longer the prediction horizon, the more the design space of the controller increases and so the more options the controller has to achieve its goals. Simulation results show that the average computational time normalized to $L = 1$ of the optimization algorithm have the following values. In the case with $L = 3, 5, 7, 9$ we have values equal to $1.28, 1.44, 1.47, 1.61$ times the time required in the case where $L = 1$. As it can be noted the longer the prediction horizon, the higher the computational complexity and the time required to solve the optimization problem.

4.4 Comparisons With Existing Methods

In this final set of experiments, we compare the proposed thermal management method with state-of-the-art convex-based thermal management techniques, in the case of time varying workload requirements. In particular, we have implemented for the comparisons the Pro-Temp technique described in [5], where we chose the frequency to satisfy the maximum workload requirements, while the maximum chip temperature is taken into account. We also implemented the technique proposed in [6]. Here we used a time horizon L equal to 9 samples(90ms). For comparison purposes, we also implemented a *threshold-based DVFS policy* (TB-DVFS), where the frequencies of the cores are matched to the application performance levels. The temperature control is here activated when a core reaches a threshold temperature level. In this case, the system reduces the maximum frequency of the system to 50% for the time-period until the next DVFS is applied. The proposed policy is implemented using time horizon L ranging from 1(10ms) to 9(90ms).

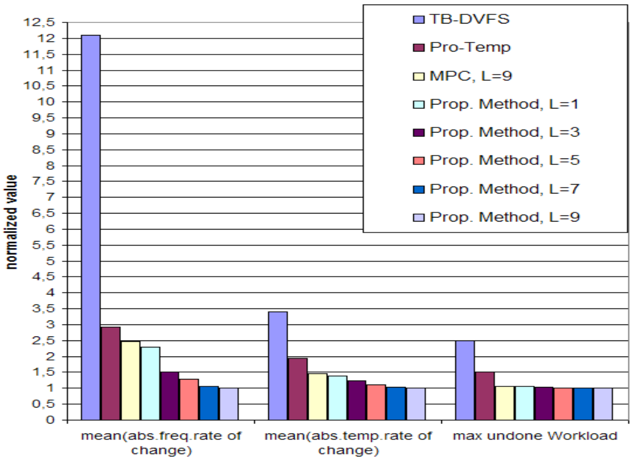

Figure 5: Statistical comparison of state-of-the-art policies normalized to the proposed method with L=9.

Figure 5, quantifies (from a statistical point of view) the improvements of the proposed policy with respect to state-of-the-art thermal management approaches. In this graph we first analyze the smoothness in both temperature and frequency variations; then we analyze the maximum value of the undone workload. To estimate the smoothness we computed the mean of the absolute value of the rate of change, with respect to the frequency settings and to the temperature profile. It can be observed how our policy outperforms previous approaches for all the previous optimality metrics for thermal management. It can be also noted by looking at the results that the proposed method outperforms also the MPC-based technique proposed in [6], even if its time horizon has been set equal to 9. The reason is because the other policies perform an optimization based on actual scheduler frequency requirements and the current and future chip thermal profile. On the contrary, the proposed policy performs an optimization also according to a prediction made on both future dynamic behaviors and future workload requirements of the system. In case of mean absolute frequency rate of change, our approach outperforms the TB-DVFS policy by a factor of $12\times$, the Pro-Temp approach by a factor of $3\times$ and the MPC-based approach by a factor of $2.5\times$. By looking at the mean absolute temperature rate of change, the proposed method outperforms by respectively $3.5\times$, $2\times$ and $1.5\times$ previously mentioned policies. Last graph compares the undone work for all the different policies. The undone workload expresses the difference between the requested workload and the workload that is actually executed by the MPSoC (see Equation 8). Despite a 5.75% reduction in the undone work of the proposed policy versus the MPC-based approach, results show respectively a $2.5\times$ improvement versus TB-DVFS and a $1.5\times$ improvement versus Pro-Temp[5].

5. CONCLUSIONS

In this work, we propose a novel thermal management policy yielding a smooth optimum control on working frequencies and voltages of multicore systems, while satisfying max-temperature and performance constraints, and minimizing power consumption. The optimization process is done by taking into account both the current thermal profile and time-varying workload requirements of the multicore system. The prediction based on the past history of the task arrival process and the online study on the reliability of the prediction is also considered in the optimization process. We compared the proposed approach with state-of-the-art thermal management techniques using as case study a commercial 8-core processing system. Results show that the proposed method achieves performance and thermal control improvements up to $12\times$.

Acknowledgment

This research has been partially funded by the Nano-Tera.ch NTF Project CMOSAIC (ref. 123618), which is financed by the Swiss Confederation and scientifically evaluated by SNSF. The authors would also like to thank the staff of the ISL laboratory in Stanford University for the interesting discussions.

6. REFERENCES

[1] P. Kongetira et al.,*Niagara: A 32-way multithreaded SPARC processor.*, IEEE Micro, 2005.

[2] K. Skadron et al.,*Temperature-aware microarchitecture: Modeling and implementation*, TACO, 2004.

[3] R. Mukherjee et al.,*Physical aware frequency selection for dynamic thermal management in multi-core systems*, Proc. ICCAD, 2006.

[4] A. Bemporad et al.,*The explicit linear quadratic regulator for constrained systems*, Automatica, 2002.

[5] S.Murali et al.,*Temperature Control of High Performance Multicore Platforms Using Convex Optimization*, Proc. DATE, 2008.

[6] F.Zanini et al. *Multicore Thermal Management with Model Predictive Control*, ECCTD 2009.

[7] J. Donald et al.,*Techniques for multi-core thermal management: Classif. and new exploration*, Proc. ISCA, 2006.

[8] G. Paci et al.,*Exploring temperature-aware design in low-power MPSoCs*, Proc. DATE, 2006.

[9] M.-N. Sabry,*High-precision compact-thermal models*. IEEE Transactions on Components and Packaging Technologies, 2005.

[10] J. Haase et al.,*Reliability-awarepower management of multi-core processors*. Proc. DIPES, 2006.

[11] JEDEC, *Failure mechanisms and models for semiconductor devices*, Jedec Solid State Tech. Association, 2003.

[12] H. Jung et al.,*Continuous Frequency Adjustment Technique Based on Dynamic Workload Prediction*, Proc. VLSI Design, 2008.

[13] A. J. Laub, et al., *Computation of System Balancing Transformations and Other Applications of Simultaneous Diagonalization Algorithms*, IEEE Trans. Automatic Control, AC-32, 1987.

[14] Coskun A.K., et al., *Proactive temperature balancing for low cost thermal management in MPSoCs*, Proc. of ICCAD 2008.

[15] Coskun A.K., et al., *Temperature management in multiprocessor SoCs using online learning*, Proc. of DAC 2008.

[16] Y. Eldar, et al., *Convex Optimization in Signal Processing and Communications*, Cambridge University Press, 2009.

[17] S. Boyd, et al., *Fast Computation of Optimal Contact Forces*, IEEE Transactions on Robotics, 23(6):1117-1132, December 2007.

[18] S. Boyd, et al., *Convex Optimization*, Cambridge University Press, 2004.

[19] M. Grant, et al., *CVX: Matlab software for disciplined convex programming*, Available at www.stanford.edu/ boyd/cvx/.