Full-Chip Runtime Error-Tolerant Thermal Estimation and Prediction for Practical Thermal Management

Hai Wang*, Sheldon X.-D. Tan*, Guangdeng Liao†, Rafael Quintanilla‡, and Ashish Gupta‡
*Department of Electrical Engineering, University of California, Riverside, CA 92521
†Department of Computer Science and Engineering, University of California, Riverside, CA 92521
‡Intel Corporation, Chandler, AZ 85226

***Abstract*—Temperature estimation and prediction are critical for on-line regulation of temperature and hot spots on today's high performance processors. In this paper, we present a new method, called FRETEP, to accurately estimate and predict the full-chip temperature at runtime under more practical conditions where we have inaccurate thermal model, less accurate power estimations and limited number of on-chip physical thermal sensors. FRETEP employs a number of new techniques to address this problem. First, we propose a new thermal sensor based error compensation method to correct the errors due to the inaccuracies in thermal model and power estimations. Second, we raise a new correlation based method for error compensation estimation with limited number of thermal sensors. Third, we optimize the compact modeling technique and integrate it into the error compensation process in order to perform the thermal estimation with error compensation at runtime. Last but not least, to enable accurate temperature prediction for the emerging predictive thermal management, we design a full-chip thermal prediction framework employing time series prediction method. Experimental results show FRETEP accurately estimates and predicts the full-chip thermal behavior with very low overhead introduced and compares very favorably with the Kalman filter based approach on standard SPEC benchmarks.**

I. Introduction

The high power density caused by the increasing integration has led to excessive temperature on chips. Although multi-core architectures partially relieve the thermal density problem, local hot spots still exist due to different loads for different cores. In order to ensure the proper working conditions of transistors and chip reliability, many dynamic thermal management (DTM) methods have been proposed, including dynamic voltage and frequency scaling (DVFS), task scheduling and computing migration [1], [2]. Recently, more effective predictive dynamic thermal management methods [3], [4], [5] have been introduced to predict the future thermal behavior and perform thermal control far before the real thermal violation occurs. However, most of the DTM methods nowadays rely only on the temperature information given by a few physical thermal sensors. To watch for temperatures in the whole chip, DTM methods have to rely on the thermal estimation based on simplified, less accurate thermal models and performance counter based power estimation, which is error-prone in practice. As a result, accurate and efficient runtime full-chip thermal estimation and prediction under those realistic and non-ideal conditions (less accurate thermal models and power estimations) are crucial for the success of practical dynamic thermal management.

Recently, many thermal modeling and simulation methods were proposed for architecture level thermal estimation [6], [7]. These methods are usually performed off-line to get the full-chip thermal behavior. Although relatively accurate, they are usually too expensive for runtime thermal estimation. Several runtime thermal estimation techniques have been studied for DTM, including the Kalman filter based methods [8], [9], the spectral based method [10], the interpolation based method [11] and the fast simulation based method [12], etc. But both the Kalman filter and fast simulation based methods are accurate only when the mean value of the power is accurately estimated by the power estimator. Both the spectral based and interpolation based methods lack the prediction capability. And the spectral based method also requires regular placement of thermal sensors.

For the runtime thermal prediction side, some methods have been proposed to predict the future temperatures, such as autoregressive moving average (ARMA) based method [13], the recursive least square based method [5] and the workload phase based method [14]. Nevertheless, these methods can only predict the temperatures at the thermal sensors and may fail to capture the potential hot spots on a chip.

In this paper, we address the practical problems of thermal estimation and prediction under the realistic conditions: inaccurate thermal model, less accurate power estimations and limited number of on-chip physical thermal sensors. The new method is called FRETEP: Full-chip Runtime Error-Tolerant Thermal Estimation and Prediction method. The main contributions of this paper are:

1) First, we propose a new thermal sensor based error compensation method to correct the errors due to the inaccuracies in thermal model and power estimations.
2) Second, we raise a new correlation based method for error compensation estimation with limited number of thermal sensors on chip. The new method explores the functional correlations of the functional blocks around each thermal sensor.
3) Third, in order to perform the thermal estimation with error compensation at runtime, compact modeling technique is optimized and integrated into the error compensation process.
4) Last, to enable accurate thermal prediction for the emerging predictive dynamic thermal management, we design a full-chip thermal prediction framework employing time series prediction method.

The rest of this paper is organized as follows. In Section II, the background of runtime thermal estimation and prediction is presented. In Section III, we demonstrate the new runtime thermal estimation and prediction method FRETEP. The experimental results are shown in Section IV. Finally, future works are given in Section V and Section VI concludes this paper.

II. Background

A. Thermal analysis in a nut shell

The heat differential equation of the chip can be spatially discretized using finite difference method in the three dimensional space

This work is supported in part by NSF grant CCF-0448534, in part by NSF Grant CCF-0902885, in part by NSF Grant CCF-0929699, in part by Semiconductor Research Corporation (SRC) grant under No. 2009-TJ-1991.

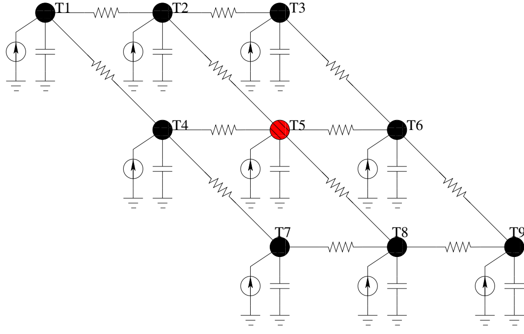


Fig. 1. A nine-grid equivalent thermal circuit. Each grid has a thermal node T_i denoted as a solid circle (black or red dashed), a thermal capacitor and a current source representing the power dissipation at the grid. There is also a thermal resistor between each pair of the adjacent thermal nodes. A thermal sensor, denoted as the red dashed circle (T_5), is placed at the center grid.

to generate an equivalent thermal circuit [15]. A two dimensional nine-grid equivalent thermal circuit example is shown in Fig. 1. As shown in the figure, each grid has a thermal node T_i, a thermal capacitor and a current source representing the power dissipation at the grid. There is also a thermal resistor between the adjacent thermal nodes. One thermal sensor, denoted as the red dashed circle, is placed at the center grid in this example.

Mathematically, if there are n discretized grids with specific boundary conditions, the equivalent thermal circuit can be modeled using an ordinary differential equation [15]

$$C\frac{dT(t)}{dt} + GT(t) = Bu(t) \tag{1}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector containing the temperatures of the n thermal nodes, $C \in \mathbb{R}^{n\times n}$ is the thermal capacitance matrix, $G \in \mathbb{R}^{n\times n}$ is the thermal conductance matrix, $B \in \mathbb{R}^{n\times p}$ is the position matrix of the input where $B_{i,j}$ denotes the portion of the jth functional block power injects into the ith thermal node and $u(t) \in \mathbb{R}^p$ contains the power dissipations of the p functional blocks. The right hand side of (1) is also written as

$$J(t) = Bu(t) \tag{2}$$

where $J(t) \in \mathbb{R}^n$ represents the power dissipations of n grids.

B. Challenges for accurate estimation and prediction

Our goal is to accurately compute all the thermal node temperatures $T(t)$ and even predict their values in the near future with small overhead. However, there are several practical problems preventing us from getting accurate temperatures in reality.

The first problem comes from the inaccurate power estimations and inaccurate thermal model. From (1), $T(t)$ can be solved numerically given the power inputs of all nodes provided by the runtime power estimator and the initial state $T(0)$. Recently, many accurate runtime functional block level power estimation methods have been proposed, for example [16], [17]. Most of these methods are based on the performance counters and are relatively accurate and computationally efficient. However, the estimated powers still contain estimation errors, especially the mean value difference, compared to the real power dissipations of the functional blocks. So the power errors will lead to inaccurate temperature estimation and prediction. Furthermore, the inaccurate thermal model is the second source of errors. Calibration can be performed to improve the accuracy, but the thermal conductivity of materials are functions of temperature, which will introduce unavoidable temperature errors if only linear thermal models are used (as the case for most existing works). In this paper, we solve this problem first in Section III-A1 by assuming sufficient number of thermal sensors.

The second problem comes from the limited number of thermal sensors on a practical chip. We further propose a correlation based method to address this problem as shown in Section III-A2.

The third problem is the computational cost issue if we want to perform the full-chip thermal estimation at runtime. In other words, it is impractical to directly work on (1) as the matrix size can be extremely large. Compact modeling technique can be used to reduce the system size, but it cannot be directly applied to the new estimation algorithm because of the newly introduced error compensation. We show in Section III-B how to optimize the compact modeling method and integrate it into the new thermal estimation method with error compensation.

Finally, a full-chip thermal prediction usually requires high computational cost because of the large thermal node number. We show in Section III-C we can still predict the full-chip temperature with small overhead with the newly designed full-chip thermal prediction framework.

III. Full-chip runtime thermal estimation and prediction method

In this section, we present the new full-chip runtime thermal estimation and prediction method FRETEP.

A. Full-chip temperature estimation

Generally, a runtime thermal estimator cannot generate accurate results due to two types of errors, as briefly introduced in Section II. One type of error is the power estimation error. Usually based on performance counters, the power estimator results may contain errors with non-zero mean and variance. Its statistics may also change at runtime because of the change of the running applications or threads. It has been shown in [12] that most part of the power is concentrated around DC and runtime power average is usually used as the input for thermal estimation. As a result, the mean value of the power estimation error is more important than the variance. The other type of thermal estimation error comes from the thermal model. There are differences in the thermal resistance and capacitance values compared to the real thermal system mainly because of the thermal effect on the thermal resistors and capacitors. We will consider both types of errors and show how accurate full-chip temperature is estimated.

Assume the inaccurate power estimation provided by the power estimator is J and the system matrices G and C are not accurate, the resulting temperature estimation is T. In order to calculate T numerically, we need to discretize (1) in time domain. Backward Euler (BE) is used here for illustration. By choosing an appropriate time step h, BE discretizes (1) in time domain as

$$(\frac{C}{h} + G)T(t+h) = \frac{C}{h}T(t) + J(t+h) \tag{3}$$

Through inverting $(\frac{C}{h} + G)$ to the right hand side, (3) is also written as

$$T(t+h) = (\frac{C}{h} + G)^{-1}(\frac{C}{h}T(t) + J(t+h)) \tag{4}$$

Given the initial value $T(0)$ and the input $J(t)$ for all time points, the subsequent temperature $T(t)$ can be calculated iteratively using (4).

However, the temperature $T(t)$ calculated from (4) is inaccurate due to the inaccurate input J and the inaccurate model G, C. Assume the actual system matrices are $\bar{G} = G + \delta G$ and $\bar{C} = C + \delta C$, and

the actual power input is $\bar{J} = J + \delta J$. The real system response $\bar{T}$ can be calculated from

$$(\frac{\bar{C}}{h} + \bar{G})\bar{T}(t+h) = \frac{\bar{C}}{h}\bar{T}(t) + \bar{J}(t+h) \quad (5)$$

1) Error compensation with sufficient thermal sensors: We would like to compensate the power estimation and model errors to generate an accurate temperature estimation.

In the ideal case, assume there are thermal sensors everywhere on the chip, that is, we have the accurate temperature information $\bar{T}(t)$ already. [1] We define the temperature estimation error δT, power estimation error δJ and model error δM as

$$\delta T(t) := \bar{T}(t) - T(t) \quad (6)$$

$$\delta J(t) := \bar{J}(t) - J(t) \quad (7)$$

$$\delta M(t) := -(\frac{\delta C}{h} + \delta G)\bar{T}(t) + \frac{\delta C}{h}\bar{T}(t-h) \quad (8)$$

Then subtract (3) from (5) and neglect the second order term, we have

$$(\frac{C}{h} + G)\delta T(t+h) = \frac{C}{h}\delta T(t) + \delta J(t+h) + \delta M(t+h) \quad (9)$$

Because of the low-pass filter property of thermal system [2], the temperature estimation error over two successive time steps does not change too much, that is $\delta T(t+h) \approx \delta T(t)$. Therefore, (9) becomes

$$(\frac{C}{h} + G)\delta T(t) \approx \frac{C}{h}\delta T(t) + \delta J(t+h) + \delta M(t+h) \quad (10)$$

We define the *error compensation term*, determined at time $t+h$, as

$$\epsilon := \delta J(t+h) + \delta M(t+h) \quad (11)$$

and from (10), the error compensation term ϵ can be approximately solved as

$$\epsilon \approx G\delta T(t) \quad (12)$$

We do not express ϵ as a variable of t since it will not be calculated repeatedly at every time point.

After we obtain the error compensation term, the inputs of all the future time points are updated as

$$J(t+ih) = J(t+ih) + \epsilon \quad (13)$$

where $i = 1, 2, \ldots$.

Note the error compensation term ϵ is accurate as long as the power estimation error statistics and the temperature do not change too much. In this case, one compensation is enough for the whole estimation time. If these conditions are not satisfied, we can perform the error compensation process (12) and (13) periodically or at the time when the temperature errors at the thermal sensors exceed a threshold.

2) Error compensation with limited number of thermal sensors: We have shown we are able to fully compensate the power estimation error and model error to generate accurate thermal estimation in the ideal case with sufficient number of thermal sensors. However, we cannot put thermal sensors all over the chip in reality. The number of sensors is always limited and as a result, it is impossible to obtain all the elements of $\delta T(t)$ in (12). In this subsection, we show how to exploit the power estimator and limited thermal sensor information and approximately recover the full-chip temperature.

[1] Note that this ideal case does not exist in reality, where there are only limited number of thermal sensors available. It is introduced here only for the purpose of better presentation and easier understanding of the realistic case shown later.

Assume there are n_s thermal sensors placed on chip. For convenience, we first perform matrix permutation on (1) to group the thermal nodes with thermal sensors together as

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \frac{dT_s(t)}{dt} \\ \frac{dT_u(t)}{dt} \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} T_s(t) \\ T_u(t) \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(t) \quad (14)$$

and

$$\begin{bmatrix} J_s(t) \\ J_u(t) \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(t) \quad (15)$$

where $T_s(t) \in \mathbb{R}^{n_s}$ represents the temperatures at the nodes where thermal sensors are placed and $T_u(t) \in \mathbb{R}^{n-n_s}$ represents the temperatures at the nodes without thermal sensors.

Accordingly, (12) becomes

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} \delta T_s(t) \\ \delta T_u(t) \end{bmatrix} = \begin{bmatrix} \epsilon_s \\ \epsilon_u \end{bmatrix} \quad (16)$$

We know the value of δT_s since thermal sensors are placed at these nodes. However, δT_u is unknown due to the absence of thermal sensors. Since there are $2n - n_s$ unknowns in (16) with n equations, (16) is unsolvable (in the normal sense) unless the number of unknowns is reduced. Fortunately, we are able to reduce the number of unknowns by taking advantage of correlation among different functional blocks in a chip.

Our idea is based on the observation that many functional blocks in a chip are highly correlated in their power consumptions. For instance, when a integer register file is busy, most likely the integer ALU and nearby cache memory will also be busy. As a result, if we properly place the thermal sensors so that more correlated functional blocks are clustered around those thermal sensors, we should be able to have a good guess of the compensation errors around the thermal sensors. Specifically, based on the placement of the n_s thermal sensors, the chip is divided into n_s blocks by combining the correlated functional blocks around each thermal sensor. We call this kind of block as *sensor block*. The compensation errors of different nodes inside one functional block are correlated and the correlation can be characterized. They are usually assumed to be the same or follow a given distribution from characterization. There are also compensation error correlations among different functional blocks inside the same sensor block, mainly because the power consumptions of these functional blocks rely strongly on a small number of common performance parameters. As a result, we introduce a *correlation matrix* $D \in \mathbb{R}^{(n-n_s)\times n_s}$ and represent ϵ_u in terms of ϵ_s as

$$\epsilon_u = D\epsilon_s \quad (17)$$

where each column of D shows the correlation of the compensation errors within a specific sensor block.

In this paper, the correlation inside the functional block is established by assuming the compensation errors are identical for all nodes. The relationship among different functional blocks inside the same sensor block is established by keeping the input ratios among different functional blocks invariant before and after the error compensation. We remark that more accurate correlation can be found through statistic characterization on realistic chips. In addition, the correlation matrix D will introduce errors if some functional blocks are not fully correlated. In this case, the errors can be minimized by placing thermal sensors properly (which is another research topic, see [18] for example) or increasing the sensor number as shown in the experiments.

We would like to walk through a simple example to illustrate this idea. As shown in Fig. 2, there are only three functional blocks on

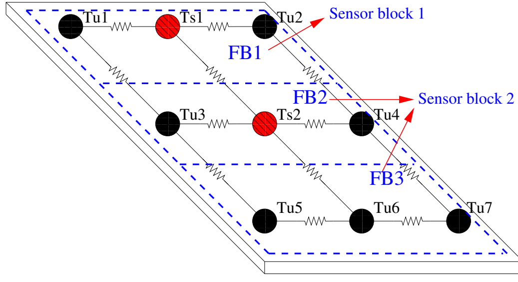


Fig. 2. A simple example with three functional blocks (FB in short in the figure) to show the D matrix construction. The functional blocks are bounded by dashed lines. The thermal sensor nodes are represented by red dashed circles and the other thermal nodes by black solid circles. The power sources and capacitors are omitted here for simplicity.

chip. Two thermal sensors (red dashed circle) are placed, one inside FB1 and the other one inside FB2. According to the two thermal sensors, the chip is divided into two sensor blocks: sensor block 1 contains FB1 and sensor block 2 includes FB2 and FB3. In this example, given ϵ_s and ϵ_u corresponding to T_s and T_u shown in Fig. 2 as

$$\epsilon_s^T = [\epsilon_{s1}\ \epsilon_{s2}] \tag{18}$$

$$\epsilon_u^T = [\epsilon_{u1}\ \epsilon_{u2}\ \epsilon_{u3}\ \epsilon_{u4}\ \epsilon_{u5}\ \epsilon_{u6}\ \epsilon_{u7}] \tag{19}$$

the D matrix can be formulated as

$$D^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \frac{J_{fb3}}{J_{fb2}} & \frac{J_{fb3}}{J_{fb2}} & \frac{J_{fb3}}{J_{fb2}} \end{bmatrix} \tag{20}$$

where J_{fb2} and J_{fb3} are the power inputs of the nodes inside FB2 and FB3 respectively. Within each functional block, the compensation errors of different nodes are the same. Take FB1 for example, there is $D_{11} = D_{21} = 1$. In order to keep the input ratios invariant among different functional blocks inside the same sensor block, the input errors of different functional blocks are set so that they are proportional to their power values. Take sensor block 2 (FB1 and FB2) for example, the input error ratio between FB2 and FB3 is J_{fb3}/J_{fb2} as shown in the second column of D. Note although J_{fb2} and J_{fb3} may change at runtime, their ratio J_{fb3}/J_{fb2} remains constant since FB2 and FB3 are correlated. We stress again that the construction of D matrix is not unique and more sophisticated characterization methods can be applied to get more accurate D.

After the introduction of the correlation matrix D, the number of unknowns has been reduced to n. Combined with (17), (16) is rearranged as

$$\begin{bmatrix} G_{12} & -I_{n_s\times n_s} \\ G_{22} & -D \end{bmatrix} \begin{bmatrix} \delta T_u(t) \\ \epsilon_s \end{bmatrix} = \begin{bmatrix} -G_{11}\delta T_s(t) \\ -G_{21}\delta T_s(t) \end{bmatrix} \tag{21}$$

where $I_{n_s\times n_s}$ is an identity matrix with dimension n_s. After ϵ_s is solved from (21) and ϵ_u is obtained from (17), the error compensation is performed with the permuted form of (13).

B. Compact modeling for fast runtime simulation

Runtime thermal estimation and prediction improve thermal management performance. However, at the same time, they introduce overhead and degrade the system performance. The overhead can be significant especially when the full-chip thermal model is used. Model order reduction (MOR) technique, which reduces the size of large dynamic system models, can be used to reduce the runtime overhead. Although MOR is a well developed technique, its integration with the new thermal estimation method is not straightforward. We will first introduce MOR very briefly (interested readers are referred to [19] for a comprehensive MOR introduction) and then show in detail how to optimize and integrate MOR into our thermal estimation method.

Assume we need to reduce the original n order model into a smaller k (usually $k \ll n$) order model. The projection based MOR method finds a projection matrix $V \in \mathbb{R}^{n\times k}$ which satisfies the following approximation

$$T(t) \approx V\tilde{T}(t) \tag{22}$$

where $\tilde{T} \in \mathbb{R}^k$ is the reduced state. In this case, the reduced model is

$$\tilde{C}\frac{d\tilde{T}(t)}{dt} + \tilde{G}\tilde{T}(t) = \tilde{B}u(t) \tag{23}$$

where $\tilde{C} = V^TCV$, $\tilde{G} = V^TGV$ and $\tilde{B} = V^TB$.

In order to integrate MOR into the new thermal estimation method with error compensation, the most important thing is to obtain the reduced error compensation term $\tilde{\epsilon}$ in the reduced model through a similar formulation of (21) and (17).

First, in order to preserve the structure of (14), the structure preserving reduction [20] is used instead of the traditional reduction method. After we get the projection matrix V of (14) using a traditional reduction method, the structure preserving reduction method divides V into two blocks according to (14) as

$$V = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{24}$$

and modify V into the structure preserving projection matrix V_{sp}

$$V_{sp} = \begin{bmatrix} \mathrm{orth}(V_1) & 0 \\ 0 & \mathrm{orth}(V_2) \end{bmatrix} \tag{25}$$

where orth means V_1 and V_2 are orthonomalized to enhance the numerical performance. In this case, the reduced model has the formulation

$$\begin{bmatrix} \tilde{C}_{11} & \tilde{C}_{12} \\ \tilde{C}_{21} & \tilde{C}_{22} \end{bmatrix} \begin{bmatrix} \frac{d\tilde{T}_s(t)}{dt} \\ \frac{d\tilde{T}_u(t)}{dt} \end{bmatrix} + \begin{bmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{bmatrix} \begin{bmatrix} \tilde{T}_s(t) \\ \tilde{T}_u(t) \end{bmatrix} = \begin{bmatrix} \tilde{B}_1 u(t) \\ \tilde{B}_2 u(t) \end{bmatrix} \tag{26}$$

where, take the G and B matrices for example, $\tilde{G}_{11} = V_1^TG_{11}V_1$, $\tilde{G}_{12} = V_1^TG_{12}V_2$, $\tilde{G}_{21} = V_2^TG_{21}V_1$, $\tilde{G}_{22} = V_2^TG_{22}V_2$, $\tilde{B}_1 = V_1^TB_1$, $\tilde{B}_2 = V_2^TB_2$.

In the reduced model, (16) becomes

$$\begin{bmatrix} \tilde{G}_{11} & \tilde{G}_{12} \\ \tilde{G}_{21} & \tilde{G}_{22} \end{bmatrix} \begin{bmatrix} \delta\tilde{T}_s(t) \\ \delta\tilde{T}_u(t) \end{bmatrix} = \begin{bmatrix} \tilde{\epsilon}_s \\ \tilde{\epsilon}_u \end{bmatrix} \tag{27}$$

where $\tilde{\epsilon}_s = V_1^T\epsilon_s$, $\tilde{\epsilon}_u = V_2^T\epsilon_u$.

There is $\epsilon_u = D\epsilon_s$ in the original model, we also need to find a similar relationship between $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ to reduce the number of unkowns in (27). Since there is an approximation

$$\epsilon_s \approx V_1\tilde{\epsilon}_s = V_1V_1^T\epsilon_s \tag{28}$$

then we have the relationship of $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ as

$$\tilde{\epsilon}_u = V_2^TD\epsilon_s \approx V_2^TDV_1\tilde{\epsilon}_s \tag{29}$$

Generally, this is not a good approximation, because $\epsilon_s = B_1\delta u + \delta M_1$ and the projection matrix V_1 may not contain the subspace spanned by B_1. In order to achieve a good approximation, V_1 in (25) is updated by appending B_1 as

$$V_1 = [V_1, B_1] \tag{30}$$

and the accurate relationship of $\tilde{\epsilon}_u$ and $\tilde{\epsilon}_s$ is

$$\tilde{\epsilon}_u \approx V_2^TDV_1\tilde{\epsilon}_s = \tilde{D}\tilde{\epsilon}_s \tag{31}$$

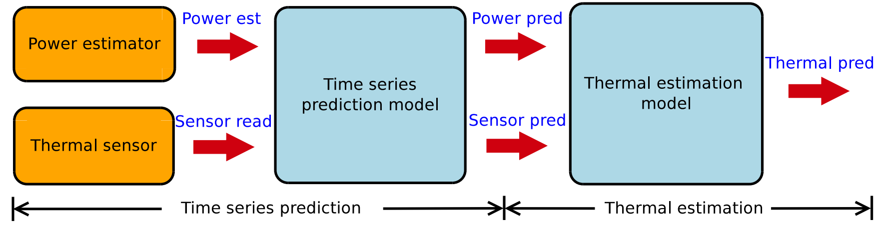


Fig. 3. Full-chip thermal prediction framework.

where
$$\tilde{D} = V_2^T D V_1 \tag{32}$$

Combining (27) and (31), $\tilde{\epsilon}_s$ is solved from the reduced version of (21) as
$$\begin{bmatrix} \tilde{G}_{12} & -I_{k\times k} \\ \tilde{G}_{22} & -\tilde{D} \end{bmatrix} \begin{bmatrix} \delta\tilde{T}_u(t) \\ \tilde{\epsilon}_s \end{bmatrix} = \begin{bmatrix} -\tilde{G}_{11}\delta\tilde{T}_s(t) \\ -\tilde{G}_{21}\delta\tilde{T}_s(t) \end{bmatrix} \tag{33}$$
and $\tilde{\epsilon}_u$ is obtained using (31).

The reduced model simulation is performed by iteratively using
$$\tilde{T}(t+h) = (\frac{\tilde{C}}{h} + \tilde{G})^{-1}(\frac{\tilde{C}}{h}\tilde{T}(t) + \tilde{J}(t+h)) \tag{34}$$
with the error compensation
$$\tilde{J}(t+ih) = \tilde{J}(t+ih) + \tilde{\epsilon} \tag{35}$$
where $i = 1, 2, \ldots$. The full-chip temperature T is recovered using (22).

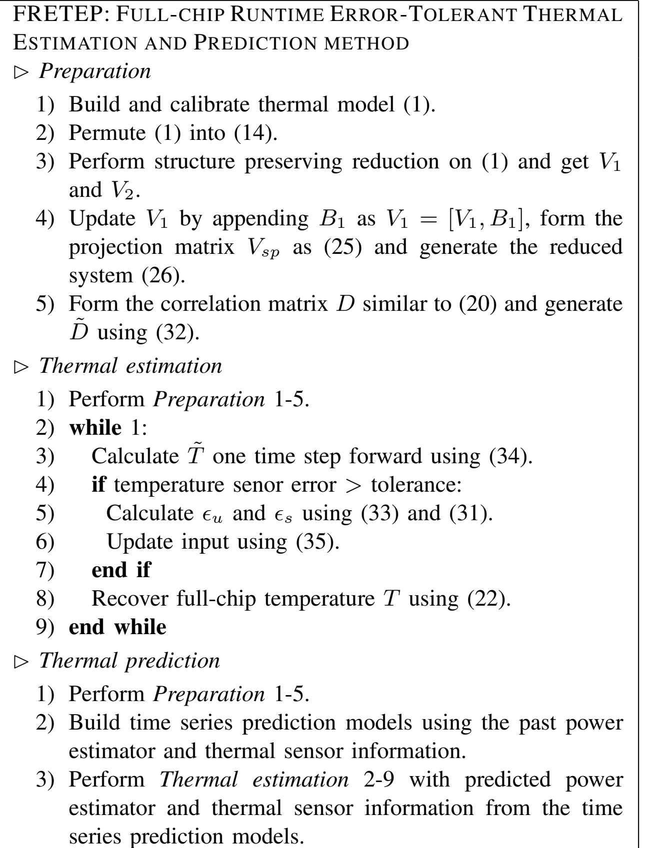
C. Full-chip runtime thermal prediction

Predicting full-chip thermal behaviors at runtime is important for the emerging predictive dynamic thermal management [3], [4], [5]. However, directly performing prediction on the estimated full-chip thermal data is very expensive due to the large number of thermal nodes, since each thermal node contains a thermal time series to be predicted. We have designed a full-chip thermal prediction framework shown in Fig. 3 by taking advantage of the small number of functional blocks, the small number of thermal sensors and the high efficiency of the newly introduced full-chip thermal estimation method. In the prediction framework, the power estimations and thermal sensor readings are first predicted using time series prediction models [21]. Then, the future full-chip temperature is calculated using the efficient thermal estimation model with the predicted power and thermal sensor information. Because the number of functional blocks (number of power estimation time series) and the number of thermal sensors (number of thermal sensor temperature reading time series) are both small (usually in dozens) compared to the number of full-chip temperature nodes (usually in thousands) and the new thermal estimation method is very efficient, the new full-chip thermal prediction framework is able to predict the full-chip thermal behaviors with small overhead.

Among many widely used time series prediction methods [21], we choose the autoregressive moving average (ARMA) model in this paper because of its long prediction range. An ARMA model consists of two parts, the autoregressive (AR) part and the moving average (MA) part. With p orders of AR part and q orders of MA part, an ARMA(p,q) model is represented as
$$y(t) + \Sigma_{i=1}^{p}(a_i y(t-i)) = e(t) + \Sigma_{i=1}^{q}(c_i e(t-i)) \tag{36}$$
where $y(t)$ is the series' value at time t (in our case, the power or thermal sensor temperature at t), $e(t)$ is the white noise term, a_i are the AR parameters and c_i are the MA parameters. Given a time frame of training data, an ARMA(p,q) model can be generated and used to predict the future data. The details of the ARMA model and other time series prediction methods can be found in [21].

D. Algorithm flow and practical considerations

The algorithm flow of FRETEP is shown in Fig. 4.


FRETEP: FULL-CHIP RUNTIME ERROR-TOLERANT THERMAL ESTIMATION AND PREDICTION METHOD

▷ *Preparation*

1) Build and calibrate thermal model (1).
2) Permute (1) into (14).
3) Perform structure preserving reduction on (1) and get V_1 and V_2.
4) Update V_1 by appending B_1 as $V_1 = [V_1, B_1]$, form the projection matrix V_{sp} as (25) and generate the reduced system (26).
5) Form the correlation matrix D similar to (20) and generate $\tilde{D}$ using (32).

▷ *Thermal estimation*

1) Perform *Preparation* 1-5.
2) **while** 1:
3) Calculate $\tilde{T}$ one time step forward using (34).
4) **if** temperature senor error $>$ tolerance:
5) Calculate ϵ_u and ϵ_s using (33) and (31).
6) Update input using (35).
7) **end if**
8) Recover full-chip temperature T using (22).
9) **end while**

▷ *Thermal prediction*

1) Perform *Preparation* 1-5.
2) Build time series prediction models using the past power estimator and thermal sensor information.
3) Perform *Thermal estimation* 2-9 with predicted power estimator and thermal sensor information from the time series prediction models.

Fig. 4. FRETEP algorithm flow.

The practical implementation of FRETEP contains off-line part and on-line part. The off-line part includes the modeling and calibration of the full thermal model (1), structure preserving reduction presented in Section III-B, and pre-factorization of $(\frac{\tilde{C}}{h} + \tilde{G})$ in the reduced model simulation (34). The on-line part contains temperature calculation using the pre-factorized $(\frac{\tilde{C}}{h} + \tilde{G})$ and error compensation. The thermal prediction contains extra on-line parts: form the ARMA model and predict temperature using the ARMA model.

TABLE I
RUNTIME AND ACCURACY COMPARISON OF FRETEP ON SPEC BENCHMARKS.

Benchmark	KF based estimation		FRETEP estimation					FRETEP prediction		
	avg err	sim time	avg err	org sim time	red sim time	X org	X kf	avg err	arma time	sim time
bzip2	3.8	0.018	0.48	0.04	0.0011	37	18	0.52	0.026	0.0011
gzip	3.9	0.006	0.36	0.14	0.0017	80	3	0.92	0.008	0.0016
mcf	3.4	0.016	0.43	0.05	0.0012	46	14	1.51	0.019	0.0011
mgrid	3.9	0.031	0.46	0.04	0.0013	34	31	0.73	0.032	0.0008
swim	4.1	0.021	0.41	0.05	0.0011	45	19	1.08	0.027	0.0013
galgel	4.5	0.008	0.37	0.11	0.0014	72	6	0.72	0.012	0.0012

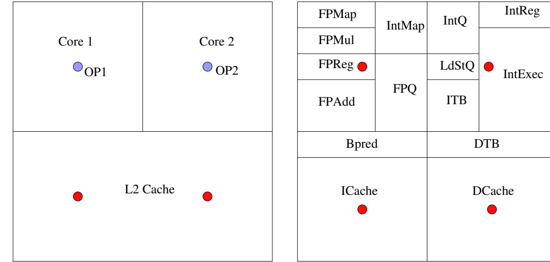
IV. EXPERIMENTAL RESULTS

The proposed method FRETEP is implemented in Matlab. All the results are collected on a Linux server with 3.0Ghz Intel Quadcore Xeon CPU and 16GB memory. In order to validate the new thermal estimation and prediction method, we build a dual-core processor with a shared L2 cache which is shown in Fig. 5 (a). The size of the processor is $10mm \times 10mm \times 0.7mm$. The core architecture shown in Fig. 5 (b) is similar to the Alpha ev6 processor. There are 10 thermal sensors placed on chip in total, 4 for each core and 2 for the L2 cache as shown in Fig. 5. We also set two observing points (OP1 and OP2) which are far away from any thermal sensors in order to demonstrate the transient estimation and prediction results. The power information is obtained using the power estimator Wattch [22] by running the standard SPEC benchmarks [23]. One core of the dual-core processor is assumed to be active and the other one is assumed to be idle, they can be switched when the temperature on one core is too high. The power estimations given by the power estimator is modeled with up to 20% mean value error. For example, the actual power and the estimated power of L2 cache of the dual-core processor by running bzip2 benchmark is shown in Fig. 6. The system model error is set to be 10%. The original order of the thermal model is 3200 and the reduced model, which is used in FRETEP, has the order of 106. The simulation time step h is chosen to be $0.1s$ to balance the speed and accuracy. The thermal estimator performs the error compensation periodically every 5 samples, that is, every 0.5 seconds. Besides FRETEP, Kalman filter based method [8] using the same reduced model as FRETEP is also implemented for comparison. For the sake of consistency, we use bzip2 benchmark to show all the transient plots and thermal map plots. All the other benchmarks show similar results and are summarized in Table I.

A. Full-chip thermal estimation results

First, we give the full-chip thermal estimation results of FRETEP using the first half of the runtime power estimator information ($0\sim 10s$). The new thermal estimator should give relatively accurate results even though the power estimation is not accurate as shown in Fig. 6. The transient estimation results for the two observing points, OP1 and OP2, are presented in Fig. 7. The error of the new thermal estimation method is within 1°C and there is no observable difference between the reduced model results and the original model results which means model order reduction is accurate enough. The Kalman filter based method has the error up to 5°C mainly because of the power estimation mean value error and the thermal model error.

In order to see the accuracy of the full-chip thermal estimation, we take a full-chip thermal map snapshot at $5s$. The results are shown in Fig. 8. It is clear that the new thermal estimation method has a thermal map very similar to the actual one while Kalman filter based method generates a thermal map with significant errors.



(a) The dual-core microprocessor architecture.

(b) The architecture for each core composed of functional blocks.

Fig. 5. The dual-core microprocessor architecture, with two cores and one shared L2 cache. 10 thermal sensors (red solid circle) are placed on chip, 2 on the L2 cache and 4 on each core. Two observing points (light blue circle) OP1 and OP2 are set in order to show the transient thermal estimation and prediction results.

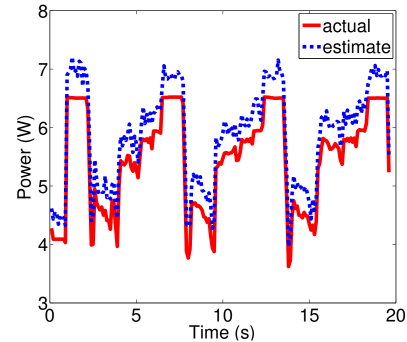


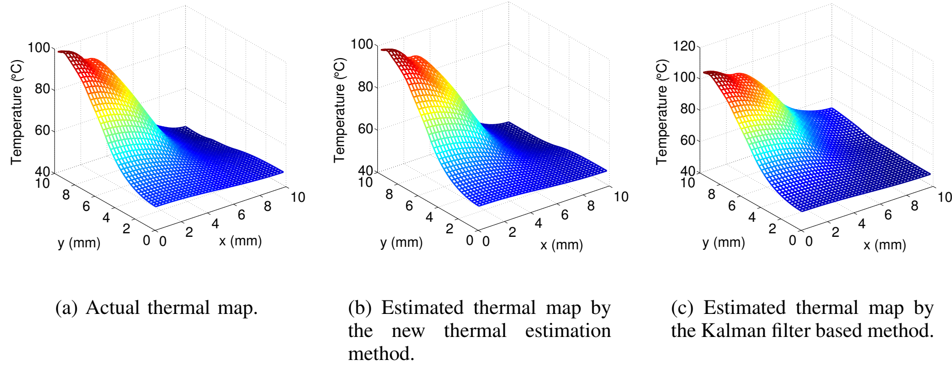
Fig. 6. The actual power and the estimated power of L2 cache running bzip2 benchmark. The estimated power has significant mean value difference compared to the actual power.

B. Full-chip thermal prediction results

Next, we demonstrate the prediction ability of FRETEP. The power estimator and thermal sensor readings are predicted by an ARMA(5, 0) model. The ARMA model is trained using 5 seconds of the past data and the prediction is performed 1 second into the future. Fig. 9 shows the prediction values of the power estimations for the L2 cache and the thermal readings from the thermal sensor on FPReg of core 1.

The accuracy of the predicted transient thermal result is shown in Fig. 10. Although the prediction errors are larger than the ones because of the errors introduced by ARMA prediction, the average error is still very small.

The comparison of the predicted thermal map snapshot at $15s$ and

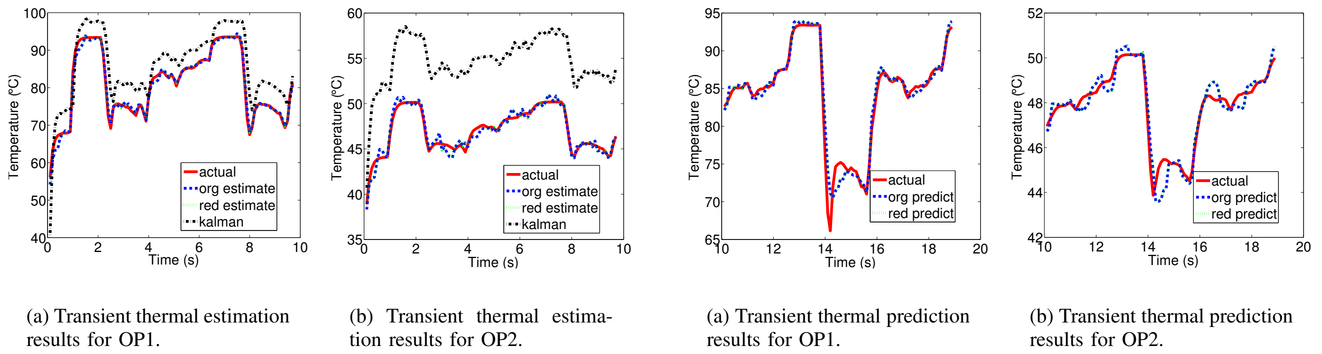


(a) Actual thermal map.

(b) Estimated thermal map by the new thermal estimation method.

(c) Estimated thermal map by the Kalman filter based method.

Fig. 8. Full-chip thermal map comparison of bzip2 benchmark at $5s$.



(a) Transient thermal estimation results for OP1.

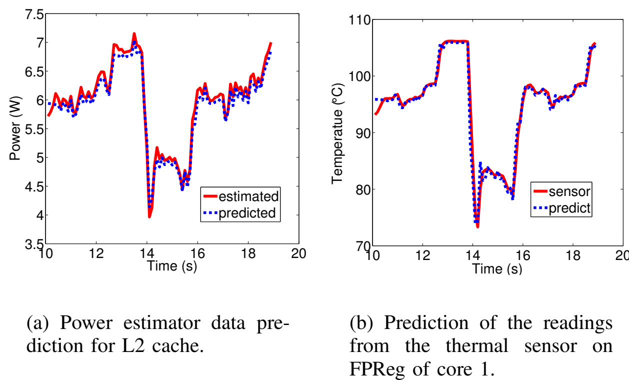
(b) Transient thermal estimation results for OP2.

(a) Transient thermal prediction results for OP1.

(b) Transient thermal prediction results for OP2.

Fig. 7. Transient thermal estimation results of bzip2 benchmark, where *org* represents the new method with original model before MOR, *red* represents the new method with reduced model after MOR and Kalman represents the Kalman filter based method [8].

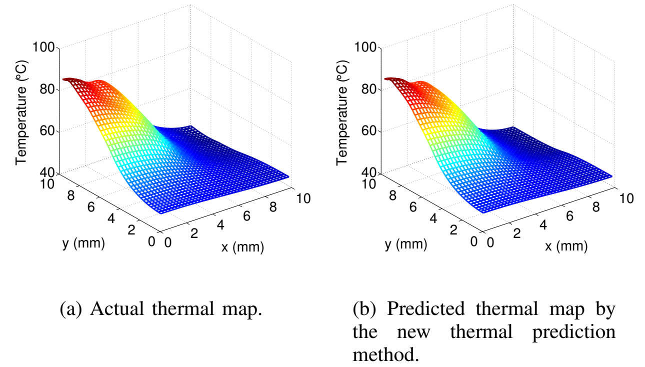
Fig. 10. Transient thermal prediction results of bzip2 benchmark where *org* represents the new method with original model before MOR and *red* represents the new method with reduced model after MOR.



(a) Power estimator data prediction for L2 cache.

(b) Prediction of the readings from the thermal sensor on FPReg of core 1.

Fig. 9. Power estimator and thermal sensor data prediction of the bzip2 benchmark.



(a) Actual thermal map.

(b) Predicted thermal map by the new thermal prediction method.

Fig. 11. Accuracy of the predicted full-chip thermal map of bzip2 benchmark at $15s$.

the actual thermal map is shown in Fig. 11. FRETEP successfully predicted the full-chip temperature.

The runtime and accuracy comparison of FRETEP on a variety of SPEC benchmarks are shown in Table I. In the table, *avg err* is the absolute error averaged on both space and time with the unit °C, *KF* means the Kalman filter based method, *X org* and *X kf* denote the speedup against the original model and the Kalman filter method with the reduced model, respectively. To be fair, all the simulation times are measured as the time spent to estimate/predict 1 second thermal behavior, with the unit s. *arma time* includes the ARMA model building time and time series prediction time. For all the benchmarks, FRETEP has better accuracy than the Kalman based method with the estimation error within 0.5°C and prediction error within 1.5°C. It also introduces very low overhead, only around 0.002

TABLE II
THERMAL SENSOR NUMBER EFFECTS ON THE ESTIMATION AND PREDICTION ACCURACIES RUNNING BZIP2.

sensor #	avg est err	avg pred err
6	0.98	0.97
8	0.58	0.61
10	0.48	0.52
14	0.42	0.44
18	0.35	0.43

seconds for 1 second estimation and 0.03 seconds for each second of prediction. It is faster than the Kalman filter based method using the same reduced model up to $20X$.

In order to study the impacts of different numbers of thermal sensor on the estimation and prediction performances, we change the sensor number from 6 to 18 and collect the average absolute errors (in $^\circ$C) in Table II. Not surprisingly, the results show increasing the number of thermal sensors reduces both the estimation and prediction errors with extra die area overhead for sensor placement.

V. FUTURE WORKS

The proposed method also contains limitations and at the same time provides new research directions. First, the accuracy of the proposed method is mainly governed by the thermal sensor placement and the correlation assumption inside the sensor blocks. A new thermal sensor placement method needs to be developed with the performance counter based power estimators in mind. Second, since the full chip thermal map is able to be accurately predicted, more effective predictive dynamic thermal management methods compared to the existing ones can be developed.

VI. CONCLUSION

In this paper, we have presented a new method FRETEP which accurately estimates and predicts the full-chip temperature at runtime under more practical conditions where we have inaccurate thermal models, less accurate power inputs and limited number of on-chip physical thermal sensors. FRETEP employs a number of new techniques to address the practical conditions problem. The proposed techniques enable FRETEP to estimate the temperature everywhere in the chip and detect the hot spots so that on-line thermal regulator can act properly to reduce the thermal gradients across the chip. Experimental results show FRETEP accurately estimates and predicts the full-chip thermal behavior with very low overhead introduced and compares very favorably with the Kalman filter based approach on standard SPEC benchmarks.

REFERENCES

[1] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proceedings of the 33rd annual international symposium on Computer Architecture*, ISCA '06, (Washington, DC, USA), pp. 78–88, IEEE Computer Society, 2006.

[2] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *International Symposium on Computer Architecture*, pp. 2–13, 2003.

[3] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore thermal management with model predictive control," in *Proc. 19th European Conference on Cicuit Theory and Design*, (Piscataway, NJ, USA), pp. 90–95, IEEE Press, 2009.

[4] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *International Symposium on Computer Architecture*, pp. 314–324, 2009.

[5] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. Design Automation Conf. (DAC)*, DAC '08, (New York, NY, USA), pp. 734–739, ACM, 2008.

[6] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, May 2006.

[7] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang, "ISAC: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 1, pp. 86–99, 2007.

[8] S. Sharifi and T. S. Rosing, "Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1586–1599, Oct. 2010.

[9] Y. Zhang and A. Srivastava, "Adaptive and autonomous thermal tracking for high performance computing systems," in *Proc. Design Automation Conf. (DAC)*, pp. 68–73, 2010.

[10] R. Cochran and S. Reda, "Spectral techniques for high-resolution thermal characterization with limited sensor data," in *Proc. Design Automation Conf. (DAC)*, pp. 478–483, 2009.

[11] J. Long, S. Ogrenci Memik, G. Memik, and R. Mukherjee, "Thermal monitoring mechanisms for chip multiprocessors," *ACM Transactions on Architecture and Code Optimization*, vol. 5, pp. 9:1–9:33, August 2008.

[12] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. yang, "Fast thermal simulation for runtime temperature tracking and management," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2882–2893, Dec. 2006.

[13] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictors for efficient thermal management in multiprocessor SoCs," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 1503–1516, October 2009.

[14] R. Cochran and S. Reda, "Consistent runtime thermal prediction and control through workload phase detection," in *Proc. Design Automation Conf. (DAC)*, pp. 62–67, 2010.

[15] Y.-K. Cheng, C.-H. Tsai, C.-C. Teng, and S.-M. Kang, *Electrothermal Analysis of VLSI Systems*. Kluwer Academic Publishers, 2000.

[16] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of MICRO*, 2003.

[17] M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi, "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters," in *International Symposium on Computer Architecture*, pp. 289–300, 2008.

[18] S. Ogrenci Memik, R. Mukherjee, M. Ni, and J. Long, "Optimizing thermal sensor allocation for microprocessors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 516–527, March 2008.

[19] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*. The Society for Industrial and Applied Mathematics (SIAM), 2005.

[20] R. W. Freund, "SPRIM: structure-preserving reduced-order interconnect macromodeling," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pp. 80–87, 2004.

[21] G. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 3rd ed., 1994.

[22] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *International Symposium on Computer Architecture*, pp. 83–94, 2000.

[23] J. L. Henning, "SPEC CPU 2000: Measuring CPU performance in the new millennium," *IEEE computer*, vol. 1, pp. 28–35, July 2000.