Parameterized Architecture-Level Dynamic Thermal Models for Multicore Microprocessors

DUO LI and SHELDON X.-D. TAN
University of California at Riverside
and
EDUARDO H. PACHECO and MURLI TIRUMALA
Intel Corporation

In this article, we propose a new architecture-level parameterized dynamic thermal behavioral modeling algorithm for emerging thermal-related design and optimization problems for high-performance multicore microprocessor design. We propose a new approach, called *ParThermPOF*, to build the parameterized thermal performance models from the given accurate architecture thermal and power information. The new method can include a number of variable parameters such as the locations of thermal sensors in a heat sink, different components (heat sink, heat spreader, core, cache, etc.), thermal conductivity of heat sink materials, etc. The method consists of two steps: first, a response surface method based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space. Second, an improved Generalized Pencil-Of-Function (GPOF) method is employed to build the transfer-function-based behavioral models for each time-varying coefficient of the polynomials generated in the first step. Experimental results on a practical quad-core microprocessor show that the generated parameterized thermal model matches the given data very well. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool *FloTHERM* on the given examples. ParThermPOF is very suitable for design space exploration and optimization where both time and system parameters need to be considered.

Categories and Subject Descriptors: J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design*

General Terms: Design, Algorithms

Some preliminary results of this article appeared in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'08)* [Li et al. 2008a].
This work is supported in part by NSF grant under No. CCF-0448534, in part by NSF grant under No. NSF Grant CCF-0902885, in part by Semiconductor Research Corporation (SRC) grant under No. 2009-TJ-1991, in part by a grant from Intel Corporation.
Authors' addresses: D. Li, S. X.-D. Tan, Department of Electrical Engineering, University of California, Riverside, CA 92521; email: {dli,stan}@ee.ucr.edu; D. H. Pacheco, M. Tirumala Intel Corporation, Hillsboro, OR 97124.

DOI 10.1145/1698759.1698766 http://doi.acm.org/10.1145/1698759.1698766

Additional Key Words and Phrases: Multicore, chip-multiprocessor, thermal modeling, architecture, behavioral modeling

ACM Reference Format:

Li, D., Tan, S. X.-D., Pacheco, E. H., and Tirumala, M. 2010. Parameterized architecture-level dynamic thermal models for multicore microprocessors. ACM Trans. Des. Autom. Electron. Syst. 15, 2, Article 16 (February 2010), 22 pages.
DOI = 10.1145/1698759.1698766 http://doi.acm.org/10.1145/1698759.1698766

1. INTRODUCTION

As CMOS technology is scaled into the nanometer region, the power density of high-performance microprocessors has increased drastically. The exponential power density increase will in turn lead to average chip temperature to raise rapidly [ITR 2007]. Higher temperature has significant adverse impacts on chip package cost, performance, and reliability. Excessive on-chip temperature leads to slower transistor speed, more leakage power consumption, higher interconnect resistance, and reduced reliability [Gunther et al. 2001; Brooks and Martonosi 2001; Pedram and Nazarian 2006].

Chip-MultiProcessing (CMP) techniques, where multiple CPU-cores and caches are integrated into a single chip, provide a viable solution to the temperature/power problems [Li et al. 2006; Advanced Micro Devices 2006; Intel 2006]. Chip-multiprocessing allows one to increase the total throughput by task-level parallel computing with lower voltage and frequency to meet power and thermal constraints. The proliferation of this technique provides both opportunities and challenges for future massive parallel computing. One difficult issue confronting designers is the unpredictable heat and thermal effects, which are caused by the placement of cores and caches and changing program loads. Furthermore, local hot spots, which may have much higher temperatures compared to the average die temperature, are becoming more prevalent in microprocessor chips [Pedram and Nazarian 2006]. This is especially the case for multicore processors as the temperature in each core can be dramatically different and the resulting large temperature gradients can produce mechanical stress and degrade the chip reliability. Hence it is very important to verify the temperatures and estimate the related performance (power, timing, yield) and reliability limits during the thermal-aware floorplanning and architecture design under various loads among different cores and caches [Skadron et al. 2003].

To facilitate this temperature-aware architecture design, it is important to have accurate and fast thermal estimation at the architecture level [Huang et al. 2005]. Both architecture and CAD tool communities are currently lacking accurate and practical tools for thermal architecture modeling. Existing work on the HotSpot project [Huang et al. 2004; Skadron et al. 2003] tried to solve this problem by generating the architecture thermal model in a bottom-up way based on the floorplanning of the function units. But this method may not be accurate for real industry designs as many approximations are made during the modeling. It may also difficult to set up for new architectures with different thermal and packaging configurations [Wu et al. 2006]. To compute the

thermal responses by solving the basic thermal transfer equations using numerical methods like the finite element method and finite difference method is very expensive, especially for different thermal conditions and package configurations during the design stage. Hence, the need for efficient, accurate, and parameterized architecture thermal models, especially for emerging multicore microprocessors, has never been greater.

Parameterized models are important for fast design exploration and optimization as one can change the parameters during the optimization process. In multicore architectures, those variable parameters related to the chip's temperature can be spatial distance of thermal sensors in the heat sink, different components in the chip package (heat sink, heat spreader, cores, cache, etc.), thermal conductivity of the heat sink materials (aluminum, copper, magnesium), or more. In addition to those parameters, the models must capture dynamic thermal behaviors. The work in Dooren et al. [2000] presents a parameterized static thermal model for flip-chip packages. The method uses a star-like resistor-only network to model the thermal behaviors of general flip-chip packages. The thermal resistors become variables of design parameters such as die size, chip thickness, etc., by means of the response surface method. Recent work [Wang et al. 2005] proposed a parameterized modeling method for performance metrics (timing, power, and temperature) at the device and gate levels using the response surface method. But this method cannot capture transient information.

In this article, we propose a new architecture-level parameterized dynamic thermal behavioral modeling algorithm for emerging thermal-related analysis and optimization problems for high-performance chip-multiprocessor design. The proposed compact thermal model will be used to predict the thermal response of new package designs once its accuracy has been calibrated and validated with the detailed models. This is the design methodology to be used by our industry partner. We propose a new approach, called ParThermPOF, to build the parameterized dynamic thermal behavioral models from accurately computed thermal and power information using the sophisticated FEA (Finite Element Analysis) or CFD (Computational Fluid Dynamics) tools at architecture level. The new method is a top-down, black-box approach, meaning it does not require any internal structure of the systems and it is very general and flexible. ParThermPOF is able to include a number of parameters such as location of thermal sensors in a heat sink, different components (heat sink, heat spreader, core, cache, etc.), thermal conductivity of heat sink materials, etc. The new method consists of two steps: first, a Response Surface Method (RSM) based on low-order polynomials is applied to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second, an improved Generalized Pencil-Of-Function (GPOF) method [Li et al. 2008b, 2009], used especifically for thermal modeling, called *ThermPOF*, is employed to build the transfer-function-based models for each time-varying coefficient of the polynomials generated in the first step.

We remark that the detailed model for generating the realistic temperatures for training compact models was developed in FloTHERM [Flotherm], which is a typical 3D Computational Fluid Dynamics (CFD) commercial software used

in cooling of electronics. Also, the material properties and boundary conditions are representatives of conditions found in the thermal test vehicles at Intel's lab rather than real-life systems. The specific workloads only mimic those of real-life applications. The model does not include the specifics of the board. The power distribution in the real die is known in advance (for example, from Thermal Test Vehicles (TTVs)). The detailed model only tries to mimic these profiles. Although sufficient detail was put in the model, the focus was to include the relevant parameters that needed to be calibrated with parameterized methods.

Simulation results on a practical quad-core microprocessor show that the generated parameterized thermal behavioral models can be built very efficiently and the temperatures computed from resulting models match the given temperatures well for given parameter space in the time domain. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool FloTHERM [Flotherm] on the given examples from our industry partner.

The rest of this article is organized as the follows: Section 2 presents the parameterized thermal modeling problem we try to solve. Section 3 explains a Generalized Pencil-Of-Function (GPOF) method for extracting poles and residues from transient responses and an improved GPOF method for thermal modeling. Section 4 presents our new parameterized thermal behavioral modeling approach based on the improved GPOF thermal modeling technique and response surface model. Section 5 shows the simulation results and Section 6 concludes this article.

2. PARAMETERIZED TRANSIENT THERMAL BEHAVIORAL MODELING PROBLEM

Two types of parameters are considered in our modeling problems. The first one is time; the second one is comparised of other parameters to be discussed shortly.

Our modeling problem is to build parameterized transient thermal models considering the both time and other variable parameters of multicore processors. Basically we want to build the behavioral model, whose inputs are the powers and outputs are temperatures that not only depend on the input powers but also depend on the system parameters. Our parameterized behavioral models are created and calibrated with the simulated temperature and power information using a commercial thermal analysis tool based on a realistic multicore processor.

In this article, we specifically look at a quad-core microprocessor architecture from Intel to validate the new thermal modeling method. The architecture of this multicore microprocessor is shown in Figure 1, where there are four CPU cores (die 0 to die 3) and one cache core (die 4). The temperatures reported are on the die bottom face and centered with each die region.

Figure 2 shows the 3D structure of this quad-core microprocessor in a package, where the CPU die (with quad-cores) is in the bottom in contact with an Intermediate Heat Spreader (IHS). At the top is the Heater Sink (HS), which

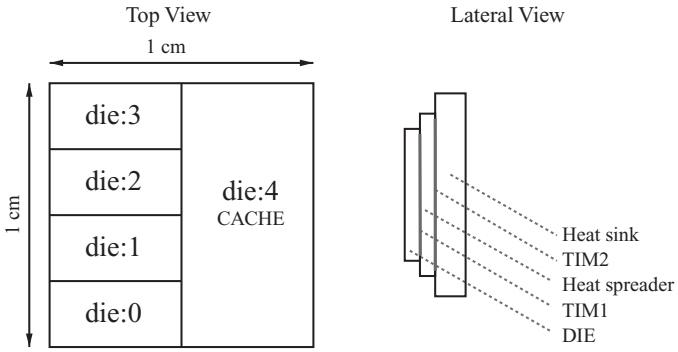


Fig. 1. Quad-core architecture.

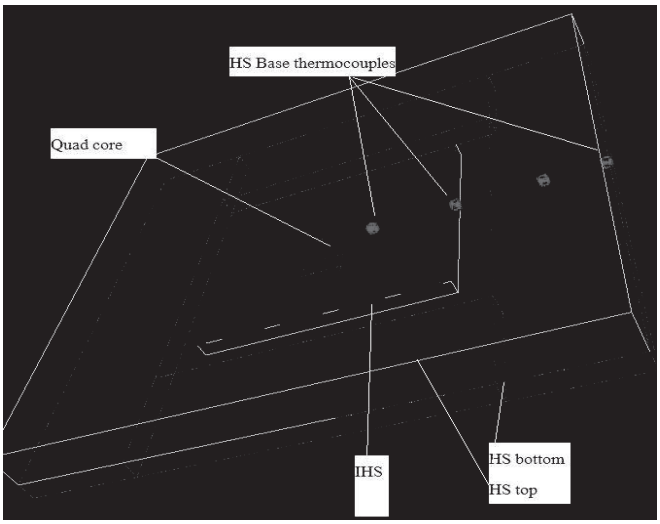
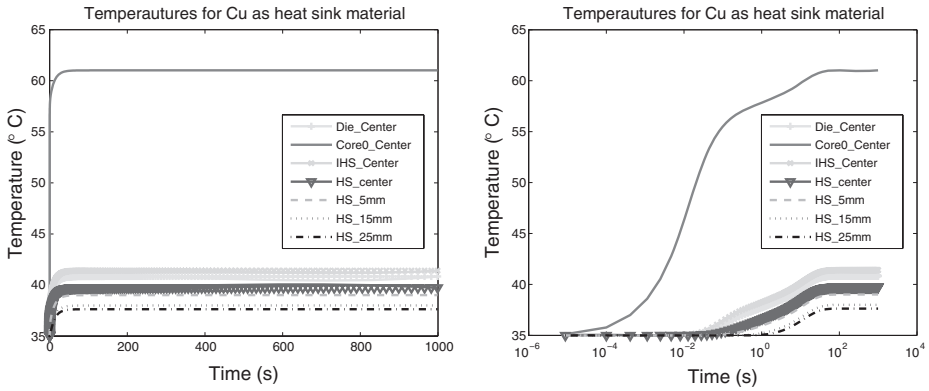


Fig. 2. 3D structure of quad-core processor.

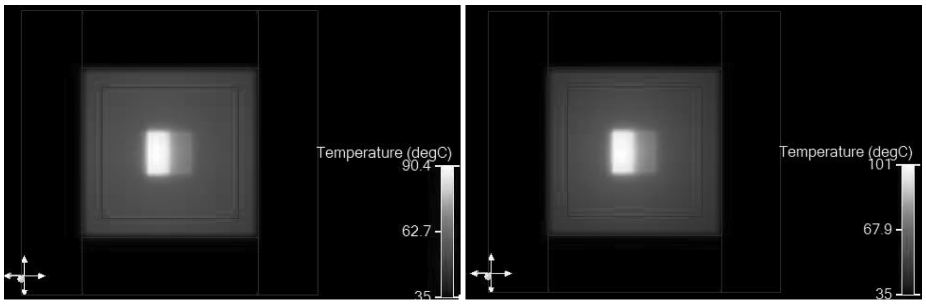
has the top and bottom parts. The thermocouples (thermal sensors) are used to measure the temperatures on these specific locations. Figure 3 shows the temperature changes when only core0 is active ($20W$ power at the input) at difference locations using a copper heat sink. HS_$5mm$ means that the temperature changes at 5mm away from the center of the heat sink. As we can see, temperatures go down as we move away from the center and away from the bottom parts of the chip. The temperatures at the core center are hottest. Also, in addition to the distance parameter in a specific component (heat sink), we may select different observation components such as individual core, cache, heat spreader, or heat sink as other indicator parameters.

Furthermore, we may consider the thermal conductivity of the heat sink material as another parameter. Normally, the heat sink is made of copper (Cu) or aluminium (Al). Cu and Al have different thermal conductivities, for example, Cu is $390W/(m \cdot K)$ and Al is $240W/(m \cdot K)$. Different heat sink materials may



(a) temperature responses in normal time scale (b) temperature responses in log time scale

Fig. 3. Temperature responses at various locations in quad-core processor when only core0 is active.



(a) using copper heat sink (b) using aluminium heat sink

Fig. 4. Temperature distributions on the whole chip with a package using different heat sink materials when all cores and cache are active.

induce the different temperature distributions on the chip. Figure 4 shows the simulated temperature distributions on the whole chip with a package using a copper heat sink and aluminum heat sink, respectively. The supply power for both cases is about 40W. We can see from the two figures that the maximum temperature of the chip on a copper heat sink is $10^{\circ}C$ less than the one on an aluminum heat sink due to different thermal conductivities of the two materials. But the prices of copper and aluminum are different. So the designers need a trade-off between hot spot temperatures and package cost. In our work, we set up such a parameter to indicate the thermal conductivity of the heat sink material properties. Such parameterized thermal models may be very helpful for design exploration and optimization.

We can abstract this quad-core processor into a linear system with 5 inputs, 1 output, and several parameters, as shown in Figure 5. The inputs are the power traces of all the cores and the output is the temperature response for given parameters. The parameters can be the location of the thermal sensors (distance to a center point), the observation component or measure point, thermal conductivity of the materials used for heat sinks, etc.

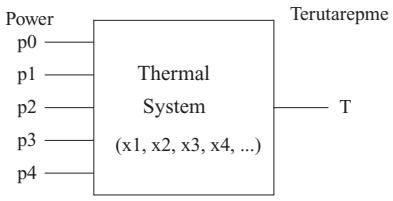


Fig. 5. Abstracted system.

Such a system can be described by the parameterized impulse-response (transfer) function matrix $\mathbf{H}$

$$\mathbf{H}(t, \xi) = [h_0(t, \xi)\ h_1(t, \xi)\ h_2(t, \xi)\ h_3(t, \xi)\ h_4(t, \xi)], \tag{1}$$

where h_i are the impulse-response functions for output due to input port i and $\xi = (\xi_1, \xi_2, \xi_3, ...)$ are the parameters of this system.

Given a power input vector for each core $\mathbf{u}(t)$ and a specific set of parameters ξ, the transient temperature can be then computed by

$$\mathbf{y}(t) = \int_0^t \mathbf{H}(t - \tau, \xi)\mathbf{u}(\tau)d\tau. \tag{2}$$

Eq. (2) can be written in the frequency domain as in (3). We have

$$\mathbf{y}(s) = \mathbf{H}(s, \xi)\mathbf{u}(s), \tag{3}$$

where $\mathbf{y}(s)$, $\mathbf{u}(s)$, and $\mathbf{H}(s, \xi)$ are the Laplace transform of $\mathbf{y}(t)$, $\mathbf{u}(t)$, and $\mathbf{H}(t, \xi)$, respectively. Each h_i can be expressed as the partial fraction form or the pole-residue form (4) [Pillage et al. 1994]

$$h_i(s, \xi) = \sum_{k=1}^{n} \frac{r_k(\xi)}{s - p_k(\xi)}, \tag{4}$$

where $h_i(s, \xi)$ is the transfer function between the ith input terminal and the output terminal; and p_k and r_k are the kth pole and residue. Once transfer functions are obtained, the transient responses can be easily computed.

To build the parameterized behavioral model, we need to solve the following two problems: (1) to find a response polynomial function that can approximate the given temperatures for all the controllable variables (parameters) with enough accuracy; (2) to find the poles and residues for each transfer function h_i from thermal coefficients (of the polynomials from step (1)) and power information to capture the transient behavior of the temperature.

For problem (1), we introduce response surface method to capture the linear or nonlinear relationship between the parameters and response (temperatures) at each time point. For problem (2), we can handle it by using the improved Generalized Pencil-Of-Function (GPOF) to extract the poles and residues from the transient thermal response. Combining (1) and (2), we can build the parameterized transient thermal behavioral models.

In the following section, we will first briefly review the improved GPOF method for transient thermal behavioral modeling before we present our new parameterized thermal behavior models.

ALGORITHM: GPOF
Input: sampling vectors $\mathbf{y}_i = [y_i, y_{i+1}, ..., y_{i+N-L-1}]^T$
Output: poles vector $\mathbf{p}$ and residues vector $\mathbf{r}$

(1) Construct matrices Y_1 and Y_2.
$Y_1 = [\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_{L-1}] \quad Y_2 = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L]$
(2) Singular value decomposition (SVD) of Y_1. $Y_1 = UDV^H$
(3) Construct matrix Z. $Z = D^{-1}U^H Y_2 V$
(4) Eigen-decomposition of Z. $Z_0 = eig(Z)$
find poles vector: $p_i = \frac{log(z_i)}{\Delta t}$
(5) Solve R_1 and R_2 from $Y_1 = Z_1 R Z_2$ and $Y_2 = Z_1 R Z_0 Z_2$.

$$Z_1 = \begin{bmatrix} 1 & 1 & ... & 1 \\ z_1 & z_2 & ... & z_M \\ \vdots & \vdots & ... & \vdots \\ z_1^{N-L-1} & z_2^{N-L-1} & ... & z_M^{N-L-1} \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 1 & z_1 & ... & z_1^{L-1} \\ & & ... & \\ \vdots & \vdots & ... & \vdots \\ 1 & z_M & ... & z_M^{L-1} \end{bmatrix}$$

find residues vector: $\mathbf{r} = \frac{R_1 + R_2}{2}$

Fig. 6. GPOF algorithm for poles and residues extraction.

3. GPOF AND IMPROVED GPOF FOR THERMAL MODELING

3.1 Review of Generalized Pencil-Of-Function (GPOF) Method

The Generalized Pencil-Of-Function (GPOF) method has been used to extract the poles and residues from the transient response of a real-time system and electromagnetism [Hua and Sarkar 1989, 1991; Sarkar et al. 1994]. It works on the sum of exponential forms, which can be expressed in the partial fraction form in frequency domain like (4). The GPOF method can be viewed as a special generalized eigenvalue computing method, in which we not only compute the eigenvalue (poles) of the given two matrices made by the sampled data, but also produce the residue for each pole in the partial fraction form [Hua et al. 2004]. As a result, it can be used to extract poles and residues from the thermal-related impulse responses for our problem. The GPOF algorithm flow is shown in Figure 6, where N is the total number of sampled points, M is the order or the number of poles, and L can be viewed as sampling window size.

For the GPOF method, it allows $M \leq L \leq N - M$, which means that we can allow different window sizes and pole numbers. Typically, choosing $L = N/2$ can yield better results.

3.2 Improved GPOF Method for Thermal Modeling

Directly applying the GPOF to the computed thermal impulse response may not lead to a stable and accurate model. We improve the GPOF method by using the logarithmic-scale and stabilization process mentioned shortly. The

resulting method, called ThermPOF, was proposed recently by the authors Li et al. [2008b, 2009]. ThermPOF builds the linear transient thermal models for given power and temperature information and is briefly reviewed in the following.

Temperatures change very rapidly in a very short time and gradually reach a steady state for a long time. This feature results in the modeling problem for the GPOF method if linear sampling is used. A logarithmic-scale sampling technique is presented in ThermPOF to mitigate this problem. After obtaining the transfer function from GPOF, ThermPOF can get the response back in the original time scale.

Also, the GPOF method will not always generate stable poles for a given impulse response. The response from the model by GPOF can be unbounded outside the sampled interval while using positive poles, although the GPOF model can give a very good matching for a given impulse response for the sampled interval. To mitigate this problem, ThermPOF artificially extends the time interval when the impulse response is zero. By sufficiently extending the time interval of zero-response in an impulse response, it can make all the poles stable.

Furthermore, the obtained impulse response may become zero numerically for a short period because temperature changes at the beginning are very slow. And long zero-response time at the beginning may cause significant discrepancies in the reduced models. To resolve this problem, ThermPOF truncates the beginning zero-response time such that the response goes to nonzero numerically immediately. The second method to mitigate this problem is by increasing the value of L, which means more sampling points but more accuracy. The advantage of the second method is that it can use the same offset for all the transfer functions, which can reduce complexity in the thermal simulation.

4. PARAMETERIZED THERMAL BEHAVIORAL MODELING METHOD

In this section, we present our new parameterized thermal behavioral modeling approach. We first present the ParThermPOF algorithm overall flow and then present the important steps in the algorithm.

4.1 The ParThermPOF Algorithm Flow

The proposed ParThermPOF consists of two major steps: first, building parameterized models by *response surface methods* on every time point; second, building subsystem response behavior models by ThermPOF. Let us assume that we have k parameter variables in the parameter space $\Omega = \{\xi | \xi = (\xi_1, \xi_2, \ldots, \xi_k)\}$.

The proposed ParThermPOF is given in Figure 7. Steps 1 and 2 build the response surface models in the parameter space Ω at each time step t. Steps 3 and 4 build the transient thermal models on top of the RSM models.

As an illustration, Figure 8 shows the response surfaces generated by ParThermPOF over 3 selected time points when only core0 is active. In the following, we discuss the important steps in the proposed method.

ALGORITHM: PARTHERMPOF
Input: the sampled transient power and temperature information in parameter space Ω
Output: parametered thermal models $T(t,P,\Omega)$

(1) Perform coding for the variable parameters where original parameter variables are converted to a new and expanded variable spaces.
(2) Obtain the coefficients of the fitting polynomials of response surface models (RSM) in a minimum square way.
1.
(3) Perform the numerical differentiation on each coefficient of the polynomials in resulting RSM and optimize the resulting waveforms for stability and accuracy.
(4) Compute the transient thermal model for each coefficient of the polynomials in resulting RSM.

Fig. 7. The proposed ParThermPOF flow.

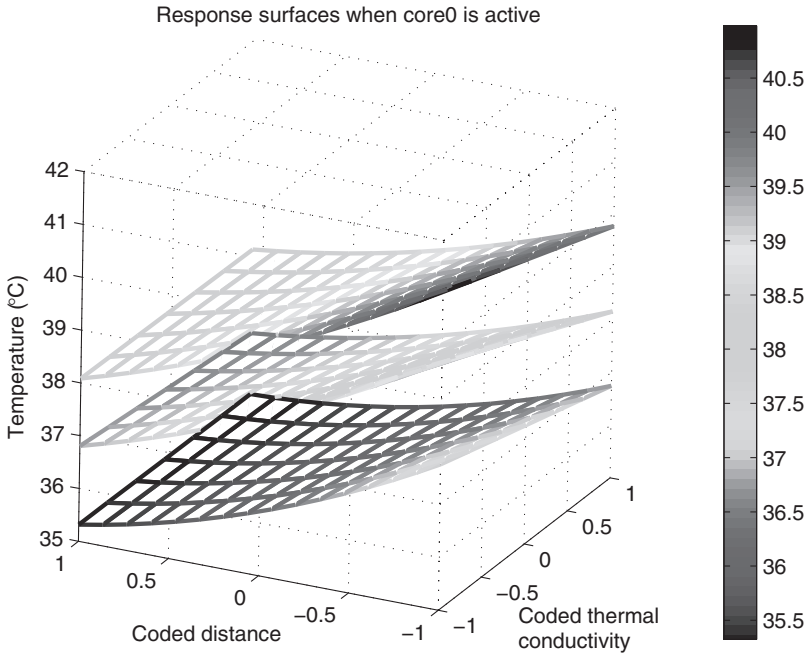


Fig. 8. Response surfaces at 3 time points when only core0 is active.

4.2 Response Surface Method

Response Surface Methodology (RSM) explores the relationships between several input variables and one or more responses. The main principle of RSM is to use a set of designed experiments to obtain an optimal response. There are many applications of RSM in real industry, particularly in situations where several input variables potentially influence some performance measure or quality characteristic of the product or process [Myers and Montgomery 2002]. This

performance measure or quality characteristic is called the response. And the input variables are sometimes called independent variables.

Specifically, suppose that a response y depends on serval controllable input variables $(\xi_1, \xi_2, \ldots, \xi_k)$

$$y = f(\xi_1, \xi_2, \ldots, \xi_k) + \varepsilon, \tag{5}$$

where the form of the true response function f is unknown and perhaps very complicated. We need to minimize the error ε when building response surface models.

Usually, a low-order polynomial in some relatively small region of the independent variable space is appropriate. In many cases, either a first-order or a second-order model is used. A first-order model is easy to estimate and apply, but it can only accurately approximate the true response surface over a relatively small region of the variable space where there is little curvature in f. But if the curvature is strong enough that the first-order model is inadequate to fit the true response surface, a second-order model will be required.

4.3 Building Parameterized Thermal Behavioral Models

4.3.1 *Coding for the Variable Parameters.* The first thing we do is to transform the natural variables ξ in a range $[a, b]$ into coded variables x in a range $[-1, 1]$. After coding, the variable matrix X will have all orthogonal columns. It may reduce numerical errors and increase numerical stability.

A simple linear transformation can be used on the original measure scale so that the highest value becomes "1" and the lowest value becomes "−1". Assume that a variable ξ_i is in a range $[a, b]$, using the linear transformation in (6), we can convert the coded variable x_i into a range $[-1, 1]$. Now we can use the coded variables $x_1, \ldots x_8$ instead of the natural ones $\xi_1, \ldots \xi_8$.

$$x_i = \frac{\xi_i - (b + a)/2}{(b - a)/2} \tag{6}$$

4.3.2 *Build the Response Surface Models.* In this article, we use a second-order response surface model. A second-order response y depending on variables $(x_1, x_2, \ldots, x_k)$ can be written as

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{j=2}^{k} \sum_{i<j} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_j x_j^2 + \varepsilon. \tag{7}$$

If we let $x_{k+1} = x_1 x_2$, $x_{k+2} = x_2 x_3, \ldots, x_{k(k+1)/2+1} = x_1^2$, $x_{k(k+1)/2+2} = x_2^2, \ldots, \beta_{k+1} = \beta_{12}$, $\beta_{k+2} = \beta_{23}, \ldots, \beta_{k(k+1)/2+1} = \beta_{11}$, $\beta_{k(k+1)/2+2} = \beta_{22}, \ldots$, then (7) becomes

$$y = \beta_0 + \sum_{j=1}^{q} \beta_j x_j + \varepsilon, \tag{8}$$

which is a linear regression model for coefficients $(\beta_0, \beta_1, \ldots, \beta_q)$, where $q = k(k + 3)/2$. We can use the least squares method to estimate the regression coefficients in the multiple linear regression model in (8).

Suppose that we have n observed responses $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ and for each y_i we have one set of parameter values $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iq})$. So (8) can be written in matrix notation as

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \tag{9}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1q} \\ 1 & x_{21} & x_{22} & \ldots & x_{2q} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{nq} \end{bmatrix}.$$
$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{10}$$

We would like to find the coefficient solution vector $\hat{\beta}$ that minimizes the squares of errors E, where

$$E = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon'\varepsilon = (\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta). \tag{11}$$

And E can be expressed as

$$\begin{aligned} E &= \mathbf{y}'\mathbf{y} - \beta'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\beta + \beta'\mathbf{X}'\mathbf{X}\beta \\ &= \mathbf{y}'\mathbf{y} - 2\beta'\mathbf{X}'\mathbf{y} + \beta'\mathbf{X}'\mathbf{X}\beta. \end{aligned} \tag{12}$$

To minimize E, we have

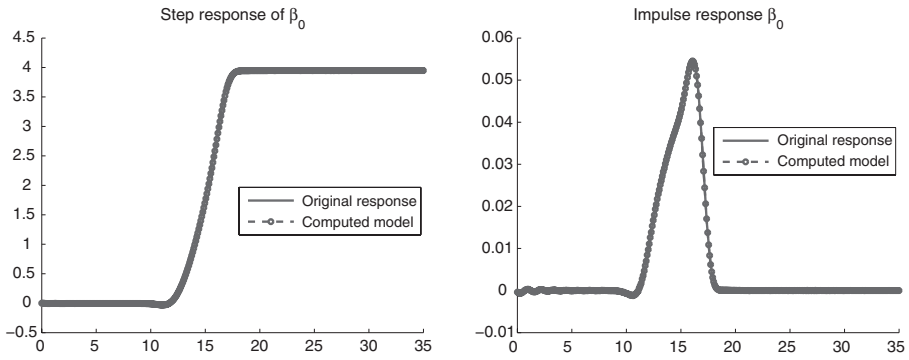
$$\left.\frac{\partial E}{\partial \beta}\right|_{\hat{\beta}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\hat{\beta} = \mathbf{0}, \tag{13}$$

so the least squares estimator of β is

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}. \tag{14}$$

In practice, we do a QR decomposition on X to make the computation numerically more stable. So we obtain $\mathbf{R}\beta = \mathbf{Q}^T\mathbf{y}$. After solving the linear equations, we get the estimated coefficient vector $\hat{\beta}$.

4.3.3 *Building Generalized Linear Thermal Models for Coefficients.* After we obtain the coded variable matrix X, the coefficients of our model can be computed using (14). At this point, we obtain the parameterized thermal model only on a single time point. We then compute the response surface models on all the time points, which could generate a set of response surface models, or more precisely, a set of coefficients, which are functions of the time now. Since we can consider the temperature response as a linear combination of such coefficients, the original thermal system is decomposed into a number of linear dynamic subsystems. Each coefficient is considered as temperature output of each subsystem and these subsystems share the same power inputs.



(a) step response of one coefficient

(b) impulse response of coefficient

Fig. 9. Step and impulse responses of one of the coefficients. The x axis is the time in logarithmic scale and y axis is the relative temperature to the ambient temperature.

To build transient models, we need to incorporate the time into our model. Now we apply the ThermPOF [Li et al. 2008b] to each coefficient, which is a function of time only and is computed from the previous RSM step. The coefficient function, which can be viewed as a special transient temperature function along with the input powers (the real temperature function is the combination of these coefficients), will become a Multiple-Input and Single-Output (MISO) linear dynamic system. In our specific thermal problem as shown in Figure 5, each coefficient function consists of 5 power inputs and 1 temperature output. Once we have the coefficient models, we can compute the total temperature response of the whole system, which is just the sum of all the responses from all the subsystems together. Note that the preceding modeling process is only for one power output. We need to repeat it 5 times in order to obtain the models of thermal system with 5 power outputs.

4.4 The Thermal-Coefficient Step and Impulse Response

In ParThermPOF, instead of having the thermal power step responses, we obtain the thermal-coefficient power step responses. Although such responses do not have direct physical meaning, the resulting step and impulse responses still resemble the thermal power step and impulse response.

Most important is that GPOF can be still applied to obtain the transient thermal models for the coefficients, which shows the flexibility of the new approach. Figure 9 shows the step and impulse responses of one of the coefficients in the resulting thermal models (versus the original ones), which are similar to the actual thermal step and impulse responses.

4.5 A Walkthrough Example

We illustrate the new method by using a real example. Specifically, the temperature response y is a function of the following parameters: two variables (ξ_1, ξ_2) are a distance away from the center and thermal conductivity of the heat sink materials; six variables $(\xi_3, \xi_4, \ldots, \xi_8)$ are used to indicate observation

components, such as core0 through core3, cache, heat spreader, and heat sink. Such variables are called indicator variables because values in them are binary (0 or 1), while values in (ξ_1, ξ_2) are continuous.

We obtained the data from Intel and the data was computed from the commercial thermal analysis tool based on a real quad-cord microprocessor. The observed temperature responses are on $\xi_1 = 0$mm, 5mm, 15mm, and $\xi_2 = 240W/(m \cdot K)$(Al), $390W/(m \cdot K)$(Cu). $\xi_3 = 1$ if we observe the temperature on core0, $\xi_4 = 1$ if we observe the temperature on core1. The settings for ξ_5, ...ξ_8 are the same. They represent core2, core3, cache, and heat spreader when they are set to 1, respectively. At any time, there is at most one variable which is set to 1 in ξ_3, ...ξ_8. When ξ_3, ...ξ_8 are all zeros, it means that we observe the heat sink.

In our setting, we set x_1 as a full second-order form, which consists of the linear terms, the crossing terms amid different variables, and squared terms. For x_2, we first consider the temperatures on two thermal conductivity points ($240W/(m \cdot K)$ and $390W/(m \cdot K)$ (we consider one more material in the experimental section). So in our models we consider x_2 as a second-order form including only linear and crossing terms. We may extend x_2 to a full second-order or even high-order form for more training data.

For $x_3, \ldots x_8$, because they are indicator variables with only binary value (0 or 1), we also consider them as a second-order form including only linear and crossing terms. Also, based on our current given data, x_3, ...x_8 only have the crossing terms with x_2, because the temperatures we obtained on 0mm, 5mm, and 15mm away from center are only for the heat sink. For other components, such as core0 through core3, cache, and heat spreader, we only know the temperature on their centers. So, currently indicator variables are independent of distance x_1. Note that we indicate the heat sink by setting $\xi_3, \ldots \xi_8$ to all zeros. So, our models can still work well to capture the temperature responses on the heat sink for different values of distance variable x_1.

Now we can begin to set up variable matrix $\mathbf{X}$ based on given temperature data like the form in (10). There are 17 terms in total, including constant, linear, crossing, and squared terms. For each time point, we have 18 given temperature samples for different distances, different thermal conductivities of heat sink materials, and different observation components. So we obtain the coded variable matrix $\mathbf{X}$ as shown in Figure 10.

4.6 More Remarks for the Proposed Method

We remark that the response surface model works fine when the response can be approximated by low-order models. Our experimental results show that for the given parameters like locations of thermal sensors in a heat sink, thermal conductivity of heat sink materials, etc., second-order approximation can give quite good approximation. For strong nonlinear parameters, new modeling techniques will be explored such as using orthogonal polynomials in RSM or piecewise linear modeling methods.

We also remark that currently the number of samples will depend exponentially on the number of variables for sufficient accuracy. We used a simple

$$
\left[\begin{array}{c|cccccccc|cccccccc}
1 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_1x_2 & x_2x_3 & x_2x_4 & x_2x_5 & x_2x_6 & x_2x_7 & x_2x_8 & x_1^2 \\ \hline
1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & -1/3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 1/9 \\
1 & -1/3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 1/9 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right] \quad (15)
$$

Fig. 10. Coded variable matrix **X**.

sampling method as sampling is not the main focus of this article. More efficient sampling methods will be investigated in the future to accommodate more parameters.

Also for fine granularity modeling where a large number of input power sources exist, the proposed method can still work. More power inputs will lead to more transfer functions and more matrix pencil operations. But the number of power inputs does not add the sampling dimensionality as all the inputs share the same time steps from one detailed simulation of a particular setting. The only difference is that each transfer function (or the coefficient function) will have more inputs.

To consider the dependency of leakage powers and thermal conductivity on temperature, the simple way is to build the thermal models on the actual measured data (we are working on this with Intel). Another way is to build the models on the detailed simulation in which such dependencies are considered. Although such a model is still linear, we at least have first-order approximation to the nonlinear effects.

In addition to the parameter variables, there are many other package variables which will affect the thermal characterization of the whole package such as the thermal conductivities of the materials used TIM1 and TIM2. In this article, we just demonstrate that the proposed method can accommodate different parameters. Our next step is to make it more practical for use in an industry setting.

5. EXPERIMENTAL RESULTS

The proposed ParThermPOF algorithm has been implemented in MATLAB 7.0 and the experimental results are obtained on a Dell PowerEdge 1900 workstation (using a Linux system) with Intel Quadcore Xeon CPUs with 2.99 Ghz and 16GB memory.

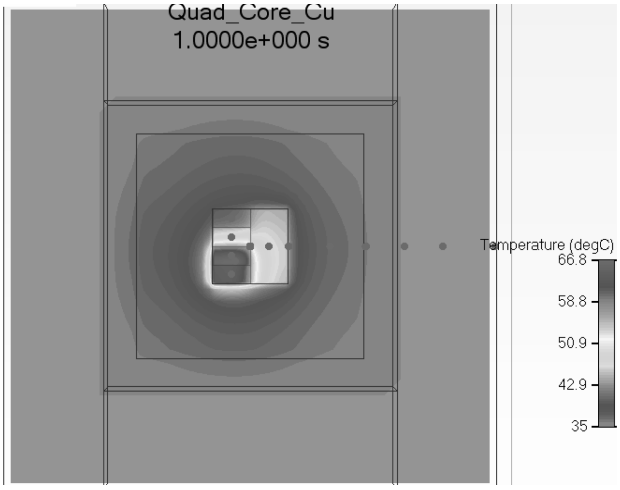


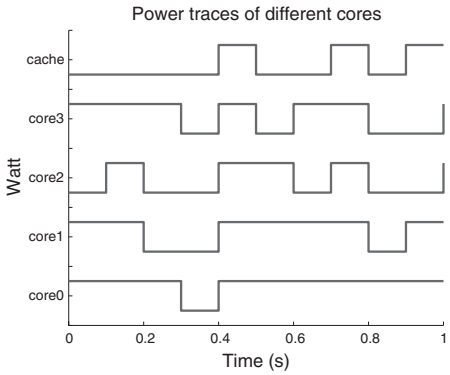
Fig. 11. Given temperature distribution on the whole chip package when using a Copper heat sink at $t = 1s$.

The example we use is the quad-core microprocessor as shown in Figure 1, from Intel. We first build parameterized thermal behavior models from a training dataset, using the commercial thermal analysis tool FloTHERM [Flotherm], which is a 3D Computational Fluid Dynamics (CFD) commercial software, where we collect the computed step temperature response when only one single step power source is applied at one time. After parameterized thermal models are built, we could apply them to generate the temperature responses for any type of time-varying input power sources and different parameter settings.

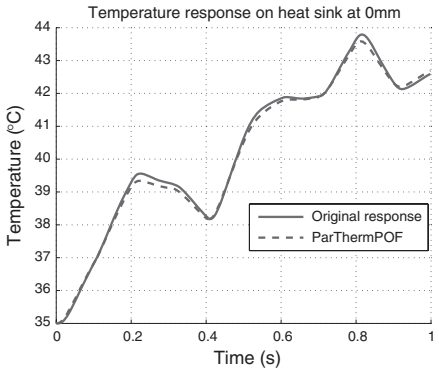
In our experiments, the training data used first to build the models have different time scales from the benchmark data used later to verify the models. Both the training data and benchmark data from Intel are the powers and computed temperatures (using FloTHERM) on a realistic quad-core microprocessor under some operating conditions. The given temperature distribution when using a Copper heat sink at $t = 1s$ is shown in Figure 11. The power input traces in the benchmark are shown in Figure 12(a), where the step power is $20W$ for all cores.

In practice, the temperature response can be computed very fast by our models during any time interval, as the computation complexity in our model is only $O(n)$ by using the recursive convolution on the pole-residue expression, where n is the number of time steps. Note that the simulation results at one time point are obtained for all the parameter space. In other words, when we change the values of parameters at one time point, the results can be computed directly without doing transient simulation again.

Now we will show the accuracy of ParThermPOF. The calculation of temperature responses at each coefficient is only done once. Then we can obtain the thermal response for any specific values for parameters $(\xi_1, \xi_2, \ldots, \xi_8)$ by setting them directly in the models.

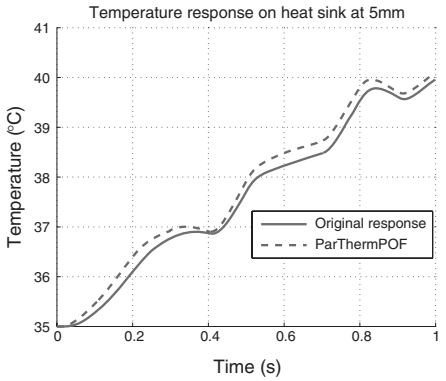


(a) power input traces in the benchmark

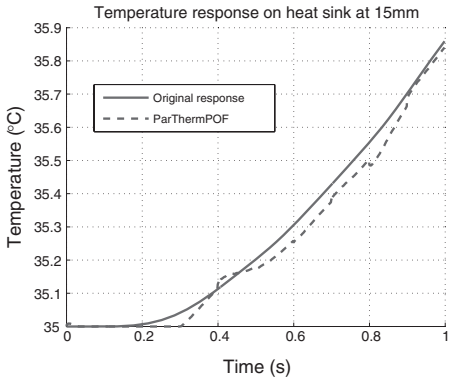


(b) temperatures on a heat sink made of Aluminum at 0mm

Fig. 12. Thermal simulation results on specific values of parameters.



(a) temperatures on heat sink made of Aluminum at 5mm



(b) temperatures on a heat sink made of Aluminum at 15mm

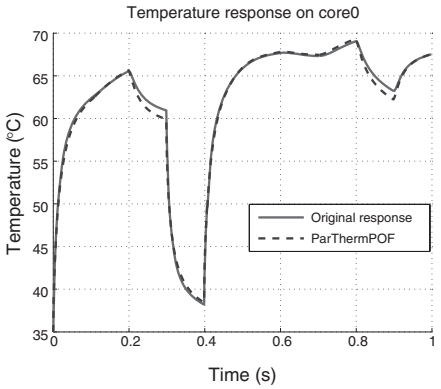
Fig. 13. Thermal simulation results on specific values of parameters.

Figure 12(b) and Figure 13 show the computed temperature results at the points 0mm, 5mm, and 15mm away from the center when using an Aluminum heat sink. In other words, we set $\xi_1 = 0, 5, 15$, $\xi_2 = 240$, and the others to zeros.

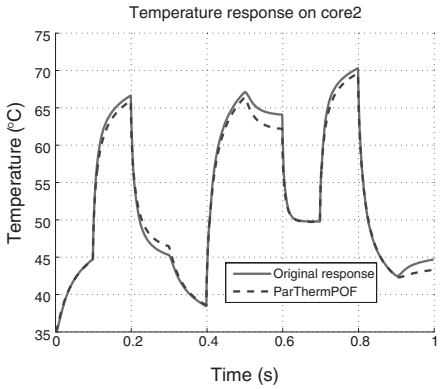
Figure 14 and Figure 15(a) show the temperatures on the center of core0, core2, and cache when using a Copper heat sink. In these cases we set $\xi_1 = 0$, $\xi_2 = 390$, $\xi_3 = 1$, or $\xi_5 = 1$, or $\xi_7 = 1$, and others to zeros.

Figure 15(b) and Figure 16 show the temperatures on the center of core1, core3, and the heat spreader when using an Aluminum heat sink. In these cases we set $\xi_1 = 0$, $\xi_2 = 240$, $\xi_4 = 1$ or $\xi_6 = 1$ or $\xi_8 = 1$, and the others to zeros.

From the figures, we can see that all the peak temperatures for each set of parameters during the whole time interval match well between computed data and given data. The models work well for the nine sets of specific parameters as

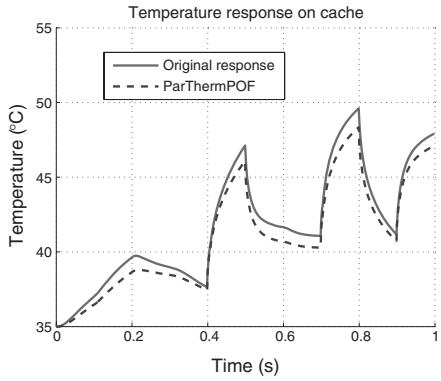


(a) temperatures on core0 when using a Copper heat sink

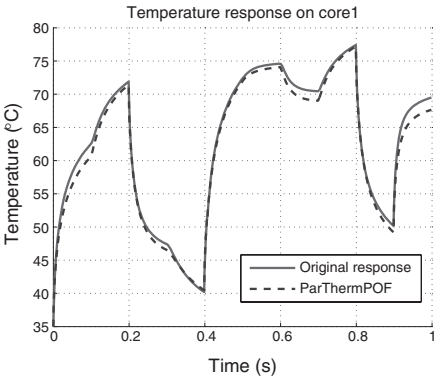


(b) temperatures on core2 when using a Copper heat sink

Fig. 14. Thermal simulation results on specific values of parameters.



(a) temperatures on cache when using a Copper heat sink



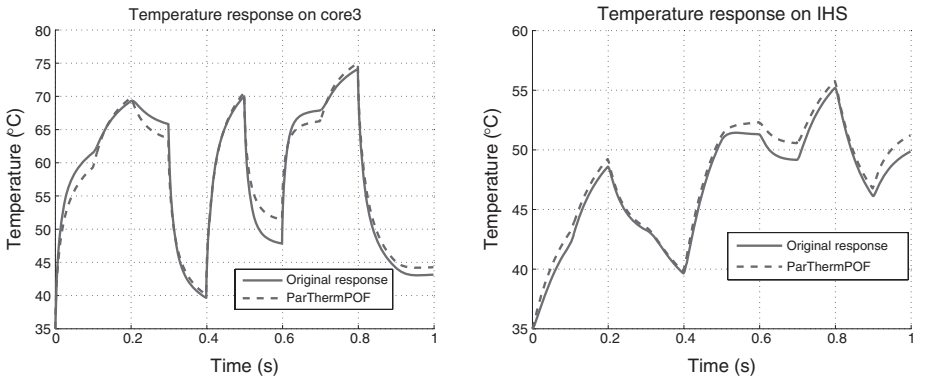
(b) temperatures on core1 when using an Aluminium heat sink

Fig. 15. Thermal simulation results on specific values of parameters.

we just showed sequentially. The errors and percentages are shown in Table I. All the temperature errors except for set6 (cache with a Copper heat sink) are less than $1^{\circ}C$.

The average errors and relative errors (computed temperature over given temperature on each time point) between computed data and given data are shown in Table II. From Table I and Table II, we can see that ParThermPOF is very accurate.

Finally, we add one sampling point for the thermal conductivity parameter of sink materials after we did for Aluminum and Copper in our model and to see how the model works. Specifically, we add the thermal-power data for Magnesium (Mg), whose thermal conductivity is $160W/(m \cdot K)$ as a heat sink material (Cu has $390W/(m \cdot K)$ and Al has $240W/(m \cdot K)$.). In this case, x_2 in



(a) temperatures on core3 when using an Aluminium heat sink

(b) temperatures on heat spreader when using an Aluminum heat sink

Fig. 16. Thermal simulation results on specific values of parameters.

Table I. Errors of the Peaks

Parameter Settings	Maximal Peak		
	Given ($^\circ C$)	Error ($^\circ C$)	Percentage
set1	43.8	0.21	0.48%
set2	40.0	0.14	0.35%
set3	35.9	0.02	0.06%
set4	69.1	0.29	0.42%
set5	70.3	0.69	0.98%
set6	49.6	1.27	2.56%
set7	77.4	0.22	0.28%
set8	74.2	0.87	1.17%
set9	55.2	0.57	1.03%

Section 4 not only includes linear and crossing terms, but also includes squared terms. We also need to add one column x_2^2 to coded variable matrix $\mathbf{X}$ and update the corresponding item values in $\mathbf{X}$.

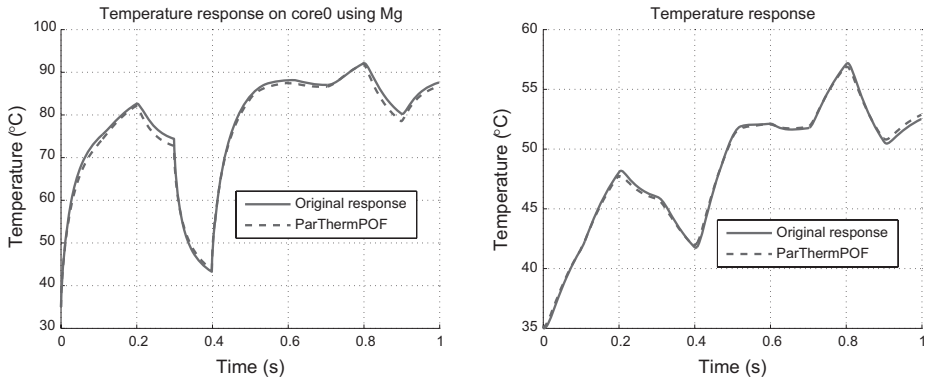
Figure 17(a) and Figure 17(b) show the temperatures on the center of core0 and heat sink when using a Magnesium heat sink when pulse-like power inputs are excited for the generated models using the new training data. In these cases we set $\xi_1 = 0$, $\xi_2 = 160$, $\xi_3 = 1$, or $\xi_3 = 0$, and the others to zeros. ParThermPOF can still obtain sufficiently accurate results.

Now we report some CPU times for the proposed method and compare them with FloTHERM [Flotherm], which uses advanced numerical techniques to compute the thermal responses. In the FloTHERM, each run for one setting (with fixed thermal materials for heat sink, heat spreader, ambient temperature, and thermal conditions) takes about 25 minutes for 1000 transient steps.

While in ParThermPOF, the training process takes 40 seconds for 5 inputs and 19 coefficients, which means it performs 5×19 matrix pencil operations [Li et al. 2008b]. After we obtain the models from training part, the simulation time

Table II. Average Errors and Relative Errors between the Computed and Given Temperatures

Parameter Settings	Average Error ($^\circ C$)	Average Relative Error ($^\circ C$)
set1	0.06	0.16%
set2	0.18	0.49%
set3	0.02	0.07%
set4	0.15	0.23%
set5	0.39	0.59%
set6	0.72	1.69%
set7	0.81	1.28%
set8	0.17	0.55%
set9	0.71	1.52%



(a) temperatures on core0 when using a Magnesium (Mg) heat sink

(b) temperatures on the center of heat sink when using a Magnesium (Mg) heat sink

Fig. 17. Thermal simulation results on specific values of parameters.

is much shorter. For the problem we have, it takes 2.81 seconds to compute the 19 coefficients for the whole simulation period (1000 steps). For one particular response at one time step, it only costs 0.002s. The reduced model has 535X speedup over FloTHERM if we only consider the transient simulation time, and 35X speedup over FloTHERM if we consider both training and transient simulation time. As a result, the proposed compact modeling is much faster than the full-blown numerical simulation.

6. CONCLUSION

In this article, we have proposed a new parameterized thermal behavioral modeling method. The new method, ParThermPOF, builds the parameterized dynamic thermal behavioral models from accurately computed thermal and power information. It can include a number of parameters such as locations of thermal sensors in a heat sink, different components (heat sink, heat spreader, core, cache, etc.), thermal conductivity of heat sink materials, etc. ParThermPOF is a general top-down, black-box parameterized performance modeling technique. It is very suitable for thermal-related design space exploration and optimization

where both dynamic behavior and system parameters need to be considered. Experimental results on a practical quad-core microprocessor have showed that the generated parameterized thermal models match the given power-thermal data very well. The compact models by ParThermPOF offer two order of magnitudes speedup over the commercial thermal analysis tool FloTHERM on the given examples.

REFERENCES

ADVANCED MICRO DEVICES. 2006. Multi-core processors - The next evolution in computing. White paper. http://multicore.amd.com.

BROOKS, D. AND MARTONOSI, M. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the International Symposium on High-Performance Computer Architecture*. 171–182.

DOOREN, S. V., BART VANDEVELDE, E. B., CHRISTIAENS, F., AND CORLATAN, D. 2000. Parametric compact models for flip chip assemblies. *IEEE Trans. Compon. Packaging Technol. 23,* 3.

FLOTHERM. http://www.flomerics.com/products/flotherm/.

GUNTHER, S., BINNS, F., CARMEAN, D., AND HALL, J. 2001. Managing the impact of increasing microprocessor power consumption. *Intel Technol. J.*

HUA, Y., GERSHMAN, A. B., AND CHENG, Q., EDS. 2004. *High-Resolution and Robust Signal Processing*. Signal Processing and Communications, vol. 19. CRC Press.

HUA, Y. AND SARKAR, T. 1989. Generalized pencil of function method for extracting poles of an em system from its transient responses. *IEEE Trans. Anten. Propag. 37*, 229–234.

HUA, Y. AND SARKAR, T. 1991. On SVD for estimating generalized eigenvalues of singular matrix pencils in noise. *IEEE Trans. Signal Process. 39,* 4, 892–900.

HUANG, W., HUMENAY, E., SKADRON, K., AND STAN, M. R. 2005. The need for a full-chip and package thermal model for thermally optimized ic designs. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'05)*. ACM, New York, 245–250.

HUANG, W., STAN, M., SKADRON, K., SANKARANARAYANAN, K., GHOSH, S., AND VELUSAMY, S. 2004. Compact thermal modeling for temperature-aware design. In *Proceedings of the Design Automation Conference (DAC)*. 878–883.

INTEL. 2006. Intel multi-core processors (white paper). http://www.intel.com/multi-core.

ITRS. 2007. International technology roadmap for semiconductors(itrs) 2007 Eds. http://public.itrs.net.

LI, D., S. X.-D. TAN, E. H. P., AND TIRUMALA, M. 2009. Architecture-Level thermal characterization for multi-core microprocessors. *IEEE Trans. VLSI. Syst. 17,* 10, 1495–1507.

LI, D., TAN, S. X.-D., PACHECO, E. H., AND TIRUMALA, M. 2008a. Parameterized transient thermal behavioral modeling for chip multiprocessors. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*. 611–617.

LI, D., TAN, S. X.-D., AND TIRUMALA, M. 2008b. Architecture-Level thermal behavioral characterization for multi-core microprocessors. In *Proceedings of the Asia South Pacific Design Automation Conference (ASPDAC)*. 456–461.

LI, Y., LEE, B. C., BROOKS, D., HU, Z., AND SKADRON, K. 2006. CMP design space exploration subject to physical constraints. In *Proceedings of the IEEE Internatiomal Symposium on High Performance Computer Architecture (HPCA)*. 15–26.

MYERS, R. H. AND MONTGOMERY, D. C. 2002. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley-Interscience.

PEDRAM, M. AND NAZARIAN, S. 2006. Thermal modeling, analysis and management in VLSI circuits: Principles and methods. *Proc. IEEE*, Special Issue on Thermal Analysis of ULSI *94,* 8, 1487–1501.

PILLAGE, L. T., ROHRER, R. A., AND VISWESWARIAH, C. 1994. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, New York.

SARKAR, T., HU, F., HUA, Y., AND WICK, M. 1994. A real-time signal processing technique for approximating a function by a sum of complex exponentials utilizing the matrix pencil approach. *Digit. Signal Process. - A Rev. J. 4,* 2, 127–140.

SKADRON, K., STAN, M., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2003. Temperature aware microarchitecture. In *Proceedings of the IEEE International Symposium on Computer Architecture (ISCA)*. 2–13.

WANG, J. M., SRINIVAS, B., MA, D., CHEN, C. C.-P., AND LI, J. 2005. System-Level power and thermal modeling and analysis by orthogonal polynomial based response surface approach (OPRS). In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 728–735.

WU, W., JIN, L., YANG, J., LIU, P., AND TAN, S. X.-D. 2006. A systematic method for functional unit power estimation in microprocessors. In *Proceedings of the Design Automation Conference (DAC)*. 554–557.

Received December 2008; revised July 2009; accepted November 2009