Parameterized Transient Thermal Behavioral Modeling For Chip Multiprocessors *

Duo Li and Sheldon X.-D. Tan
Dept. of Electrical Engineering
University of California
Riverside, CA 92521

Eduardo H. Pacheco and Murli Tirumala
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052

ABSTRACT

In this paper, we propose a new architecture-level parameterized transient thermal behavioral modeling algorithm for emerging thermal related design and optimization problems for high-performance chip-multiprocessor (CMP) design. We propose a new approach, called *ParThermPOF*, to build the parameterized thermal performance models from the given architecture thermal and power information. The new method can include a number of parameters such as the locations of thermal sensors in a heat sink, different components (heat sink, heat spread, core, cache, etc.), thermal conductivity of heat sink materials, etc. The method consists of two steps: first, 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 matchs the given data very well. *ParThermPOF* is very suitable for design space exploration and optimization where both time and system parameters need to be considered.

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 [2]. Higher temperature has significant adverse impacts on chip packing cost, performance and reliability. Excessive on-chip temperature leads to slower transistor speed, more leakage power consumption, higher interconnect resistance, and reduced reliability [4, 5, 12].

Chip-multiprocessing techniques, where multiple CPU-cores and caches are integrated into a single chip, provide a viable solution to the temperature/power problems [1,3,10]. 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 the designers is the unpredictable heat and thermal effects, which are caused by the placement of cores and caches and changing program loads. So it is very important to verify the temperatures during the floorplanning and architecture design under various loads among different cores and caches. The estimated temperature at the architecture level can then be used to perform power, performance, and reliability analysis, together with floorplanning and packaging design [14].

To facilitate this temperature-aware architecture design, it is important to have accurate and fast thermal estimation at the architecture level. Both architecture and CAD tool communities are currently lacking reliable and practical tools for thermal architecture modeling. Existing work on the HotSpot project [8, 14] 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 is difficult to set up for new architecture with different thermal and packaging configurations [16]. Also the resulting models work for only traditional one-core CPU architecture and the accuracy may be not sufficient as many approximations are made. And this method cannot accommodate variable parameters.

Parameterized models are important for fast design exploration and optimization as one can change the parameters during the optimization process. In the multi-core architecture, 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 (aluminium and copper) or more. In addition to those parameters, the models must capture transient thermal behaviors. Recent work [15] proposed a parameterized modeling method for performance metrics (timing, power, and temperature) at the device and gate levels using response surface method. But this method cannot capture the transient information.

In this paper, we propose a new architecture-level parameterized transient thermal behavioral modeling algorithm for emerging thermal related analysis and optimization problems for high-performance chip-multiprocessor design. We propose a new approach, called *ParThermPOF*, to build the parameterized transient thermal behavioral models from the measured or computed thermal and power information at the 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 spread, core, cache, etc.), thermal conductivity of heat sink materials, etc. The new method consists of two steps: first, 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 [9], specifically 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.

*This work is supported in part by NSF CAREER Award CCF-0448534, NSF Grant CCF-0541456, UC Micro Program #07-101 via Intel Corp.

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

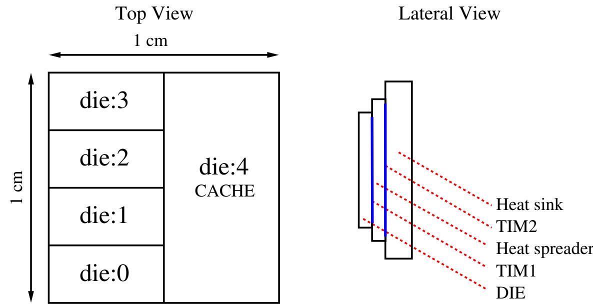


Figure 1: Quad-core architecture

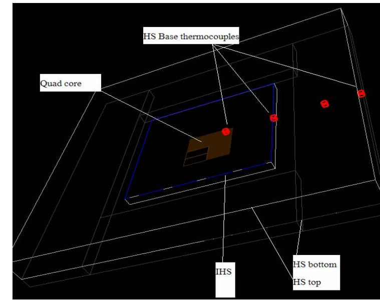


Figure 2: 3D structure of quad-core processor

Simulation results on a practical quad-core microprocessor show that the generated parameterized thermal behavioral models can be built very efficiently and the resulting models match the measured temperatures well for given parameter space in time domain.

The rest of this paper is organized as the follows: Section 2 presents 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 response 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 paper.

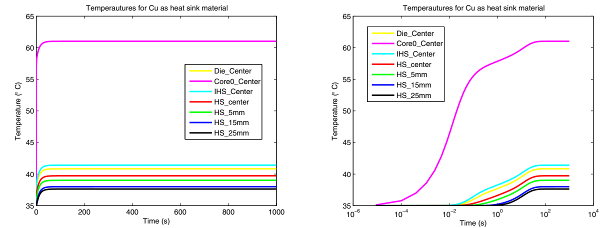
2. PARAMETERIZED TRANSIENT THERMAL BEHAVIORAL MODELING PROBLEM

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

Our modeling problem is to build parameterized transient thermal models considering the both time and other variable parameters of multi-core 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 measured/simulated temperature and power information from the real chips.

In this paper, we specifically look at a quad-core microprocessor architecture from Intel to validate the new thermal modeling method. The architecture of this multi-core microprocessor is shown in Fig. 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.

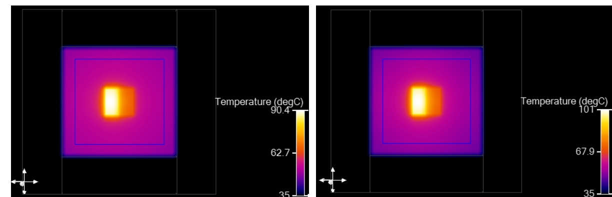
Fig. 2 shows 3-D structure of this quad-core microprocessor, where the CPU die (with quad-cores) is in the bottom in contact with intermediate heat spreader (IHS) in blue. At the top is the heater sink (HS), which has the top and bottom parts. The thermocouples (thermal sensors) in red are used to measure the real temperatures on these specific locations. Fig. 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



(a) Temperature responses in normal time scale.

(b) Temperature responses in log time scale

Figure 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

Figure 4: Temperature distributions on the whole chip using different heat sink materials when all cores and cache are active.

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, heat sink is made of copper (Cu) or aluminium (Al). Cu and Al have different thermal conductivities, such as Cu is $390W/(m \cdot K)$ and Al is $240W/(m \cdot K)$. Different heat sink materials may induce the different temperature distributions on the chip. Fig. 4 shows the temperature distributions on the whole chip using copper heat sink and aluminium heat sink. We can see from the two figures that the maximum temperature of the chip on a copper heat sink is $10C^{\circ}$ less than the one on an aluminium heat sink. But the price of copper heat sink is apparently much higher than the aluminum one. 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 the 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 Fig. 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.

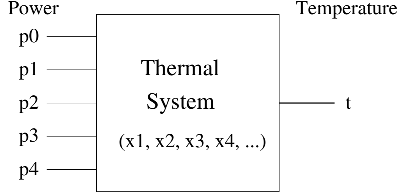


Figure 5: Abstracted system

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

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

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

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

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

Equation (2) can be written in frequency domain as in (3).

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

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).

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

where $h_i(s,\xi)$ is the transfer function between the ith input terminal and the output terminal; 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 functions that can approximate the measured 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;

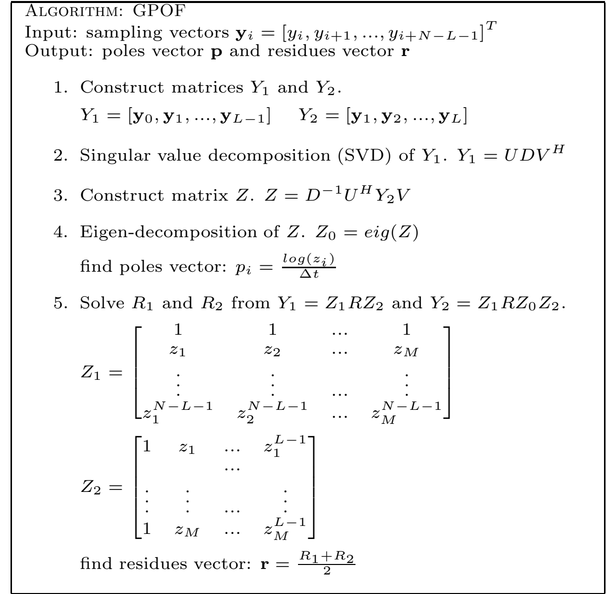
For problem (1), we introduce response surface method to capture linear or nonlinear relationship between the parameters and response (temperatures) at each time point. For problem (2), we can handle it by using improved generalized pencil-of-function (GPOF) to extract the poles and residues from transient thermal response. Combine (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.

3. GPOF AND IMPROVED GPOF FOR THERMAL MODELING

3.1 Review of generalized pencil-of-function (GPOF) method

Generalized pencil-of-function (GPOF) method can be used to extract the poles and residues from the transient response of a real-time system and electromagnetism [6,7,13]. It works on the sum of exponential forms, which can be expressed in the partial fraction form in frequency domain like (4). So it can be used to extract poles and residues from the impulse responses for our problem. GPOF algorithm flow can be shown in Fig 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.



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 $Y_2 = Z_1 R Z_0 Z_2$.
   $$Z_1 = \begin{bmatrix} 1 & 1 & \dots & 1 \\ z_1 & z_2 & \dots & z_M \\ \vdots & \vdots & \dots & \vdots \\ z_1^{N-L-1} & z_2^{N-L-1} & \dots & z_M^{N-L-1} \end{bmatrix}$$
   $$Z_2 = \begin{bmatrix} 1 & z_1 & \dots & z_1^{L-1} \\ & & \dots & \\ \vdots & \vdots & \dots & \vdots \\ 1 & z_M & \dots & z_M^{L-1} \end{bmatrix}$$
   find residues vector: $\mathbf{r} = \frac{R_1 + R_2}{2}$

Figure 6: GPOF algorithm for poles and residues extraction

For GPOF method, it allows $M \leq L \leq N - M$, which means that we can allow the different window size 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 stable and accurate model. We improve the GPOF method by using logarithmic-scale and stabilization process mentioned below. The resulting method, called *ThermPOF*, was proposed recently by the authors [9]. *ThermPOF* builds the linear transient thermal models for given power and temperature information and is briefly reviewed in the following.

Because the measured 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 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 original time scale.

Also, 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 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.

Further more, the obtained impulse response may become zero numerically for a short period because temperature changes at the beginning is very slow. And long zero-response time at beginning may cause the significant discrepancies in the reduced models. To resolve this problem, *ThermPOF* truncates the beginning zero-response time such that response goes to non-zero 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 the complexity in the thermal simulation.

4. PARAMETERIZED THERMAL BEHAVIORAL MODELING METHOD

In this section, we present our new parameterized thermal behavioral modeling approach based on the *ThermPOF* method and response surface method.

4.1 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 [11]. This performance measure or quality characteristic is called response. And the input variables are sometimes called independent variables.

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

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

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. 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.

4.2 Building response surface model for thermal response

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 the numerical errors and increase the numerical stability.

A second-order response y depending on variables $(x_1, x_2, ..., x_k)$ can be written as

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{j=2}^{k} \sum_{i<j} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_j x_j^2 + \varepsilon \tag{6}$$

If we let $x_{k+1} = x_1 x_2$, $x_{k+2} = x_2 x_3$, ... ,$x_{k(k+1)/2+1} = x_1^2$, $x_{k(k+1)/2+2} = x_2^2$, ... , $\beta_{k+1} = \beta_{12}$, $\beta_{k+2} = \beta_{23}$, ... ,$\beta_{k(k+1)/2+1} = \beta_{11}$, $\beta_{k(k+1)/2+2} = \beta_{22}$, ..., then (6) becomes

$$y = \beta_0 + \sum_{j=1}^{q} \beta_j x_j + \varepsilon \tag{7}$$

which is a linear regression model for coefficients $(\beta_0, \beta_1, ..., \beta_q)$, where $q = k(k+3)/2$. We can use least squares method to estimate the regression coefficients in the multiple linear regression model in (7).

Suppose that we have n observed responses $\mathbf{y} = (y_1, y_2, ..., y_n)$ and for each y_i we have one set of parameter values $\mathbf{x_i} = (x_{i1}, x_{i2}, ..., x_{iq})$. So (7) can be written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \varepsilon \tag{8}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1q} \\ 1 & x_{21} & x_{22} & \dots & x_{2q} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nq} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_q \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{9}$$

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

$$E = \sum_{i=1}^{n} \varepsilon_i^2 = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \tag{10}$$

And E can be expressed as

$$\begin{aligned} E &= \mathbf{y}'\mathbf{y} - \boldsymbol{\beta}'\mathbf{X}'\mathbf{y} - \mathbf{y}'\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \\ &= \mathbf{y}'\mathbf{y} - 2\boldsymbol{\beta}'\mathbf{X}'\mathbf{y} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \end{aligned} \tag{11}$$

To minimize E, we have

$$\left.\frac{\partial E}{\partial \boldsymbol{\beta}}\right|_{\hat{\boldsymbol{\beta}}} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{0} \tag{12}$$

So the least squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{13}$$

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

4.3 Building parameterized thermal behavioral models

In this paper, we use second-order response surface model. We illustrate the new method by using a real example.

4.3.1 Coding for the variable parameters

Specifically, the temperature response y is function of the following parameters: two variables (ξ_1, ξ_2) are distance away from the center and thermal conductivity of the heat sink materials; six variables $(\xi_3, \xi_4, ..., \xi_8)$ are used to indicate observation components, such as core0-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 quantitive.

We obtained the data from Intel and the data was measured by the thermal sensors from a real quad-cord microprocessor. The observed temperature responses are on $\xi_1 = 0mm$, $5mm$, $15mm$ 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 are on core1. The setting for $\xi_5, ...\xi_8$ 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 $\xi_3, ...\xi_8$. When $\xi_3, ...\xi_8$ are all zeros, it means that we observe on heat sink.

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 quantitive variable ξ_i is in a range $[a, b]$, using the linear transformation in (14), we can convert the coded variable x_i into a range $[-1, 1]$. Now we can use the coded variables $x_1, ...x_8$ instead of the natural ones $\xi_1, ...\xi_8$.

$$x_i = \frac{\xi_i - (b+a)/2}{(b-a)/2} \tag{14}$$

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. However, for x_2, based on the current data we obtained, we only have the measured temperatures on two thermal conductivity points ($240W/(m \cdot K)$ and $390W/(m \cdot K)$. So in our models we consider x_2 as a second-order form including only linear and crossing terms. In the future, after obtaining more measured data collected by Intel, we may extend x_2 to a full second-order or even high-order form.

For $x_3, ...x_8$, because they are indicator variables 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 measured data, $x_3, ...x_8$ only have the crossing terms with x_2. Because the temperatures we measured on $0mm$, $5mm$ and $15mm$ away from center is only for heat sink. For other components, such as core0-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 $\xi_3, ...\xi_8$ to all zeros. So our models can still work well to capture the temperature responses on heat sink for different values of distance variable x_1.

Now we can begin to set up variable matrix $\mathbf{X}$ based on measured temperature data like the form in (9). There are 17 terms in total, including constant, linear, crossing and squared terms. For each time point, we have 18 measured 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 Fig. 7.

4.3.2 Building generalized linear thermal models for coefficients

After we obtain coded variable matrix X, the coefficients of our model can be computed using (13). 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 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.

To build transient models, we need to incorporate the time into our model. Now we apply the *ThermPOF* [9] to each coefficient, which is computed from the RSM and is function of time now. The coefficient, a special temperature, along with the input powers, will become single output and multiple inputs (5 inputs) system based on our thermal models as shown in Fig. 5, which consists of 5 power inputs. Once we have the model, we can compute the total temperature

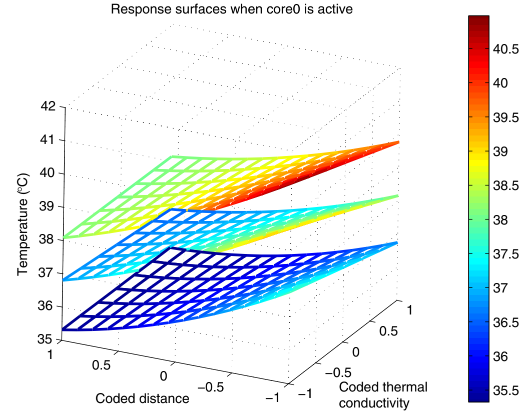


Figure 8: Response surfaces at 3 time points when only core0 is active

response of the whole system, which is just the sum of all the responses from all the subsystems together. Note that the modeling process above is only for one power input. We need to repeat it 5 times in order to obtain the models of thermal system with 5 power inputs.

Overall, *ParThermPOF* consists of two steps: first, building parameterized models by response surface methods on every time point; second, building subsystem response behavior models by *ThermPOF*. Fig. 8 shows the response surfaces generated by *ParThermPOF* over 3 selected time points when only core0 is active.

5. SIMULATION RESULTS

The proposed *ParThermPOF* algorithm has been implemented in MATLAB 7.0 and tested on the quad-core microprocessor architecture, as shown in Fig. 1, from Intel. We first build parameterized thermal behavior models from a training data set where we collect the measured temperatures 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 are provided by Intel who measured the temperatures on a real quad-core microprocessor under real operating conditions. The measured temperature distribution when using a copper heat sink at $t = 1s$ is shown in Fig. 9. The power input traces in the benchmark are shown in Fig. 10(a), where the step power is $20W$ for all the cores.

In practice, 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 recursive convolution on the pole-residue expression, where n is the number of time steps. Note that the simulation results are also subject to parameter changes. When we change the values of parameters, the results can be computed directly without doing 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 thermal response for any specific values for parameters $(\xi_1, \xi_2, ..., \xi_8)$ by setting them directly in the models.

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

Figure 7: Coded variable matrix X

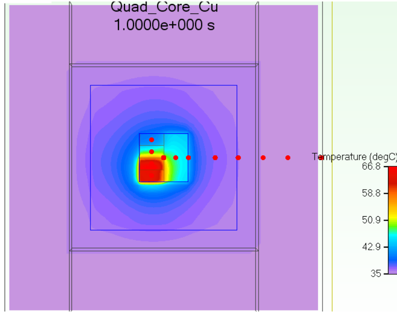
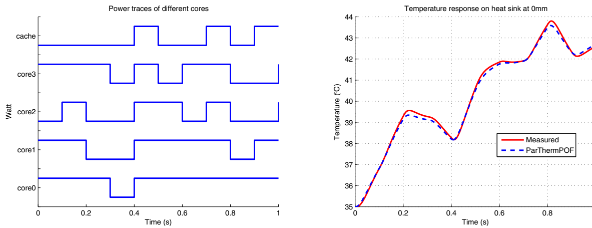


Figure 9: Measured temperature distribution on the whole chip when using a copper heat sink at $t = 1s$.



(a) Power input traces in the benchmark.

(b) Temperatures on a heat sink made of aluminium at 0mm.

Figure 10: Thermal simulation results on specific values of parameters

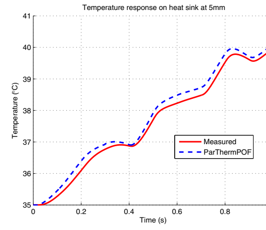
Fig. 10(b) and Fig. 11 show the computed temperature results at the points $0mm$, $5mm$ and $15mm$ away from the center when using an aluminium heat sink. In another word, we set $\xi_1 = 0, 5, 15$, $\xi_2 = 240$ and others to zeros.

Fig. 12 and Fig. 13(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.

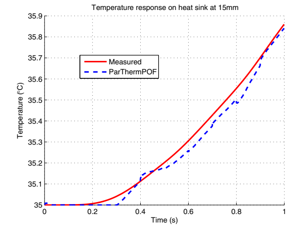
Fig. 13(b) and Fig. 14 show the temperatures on the center of core1, core3 and the heat spreader when using an aluminium heat sink. In these cases we set $\xi_1 = 0$, $\xi_2 = 240$, $\xi_4 = 1$ or $\xi_6 = 1$ or $\xi_8 = 1$ and others to zeros.

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 measured data. The models work well for the nine sets of specific parameters as we just showed sequentially. The errors and percentages are shown in Table 1. All the temperature errors except for set6 (cache with a copper heat sink) is less than $1^{\circ}C$.

The average errors and relative errors (computed temperature over measured temperature on each time point) between computed data and measured data are shown in Table 2. From Table 1 and Table 2, we can see that *ParThermPOF* is very accurate.

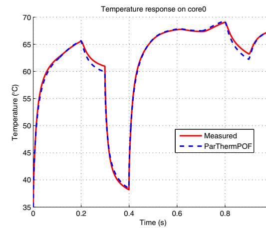


(a) Temperatures on heat sink made of aluminium at 5mm.

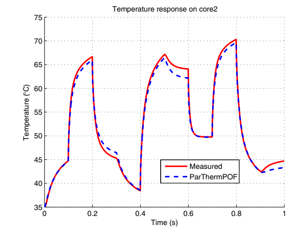


(b) Temperatures on a heat sink made of aluminium at 15mm.

Figure 11: Thermal simulation results on specific values of parameters



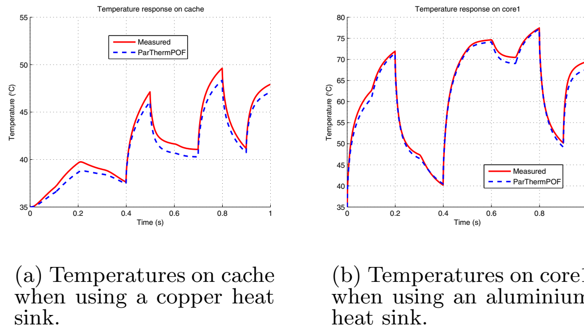
(a) Temperatures on core0 when using a copper heat sink.



(b) Temperatures on core2 when using a copper heat sink.

Figure 12: Thermal simulation results on specific values of parameters

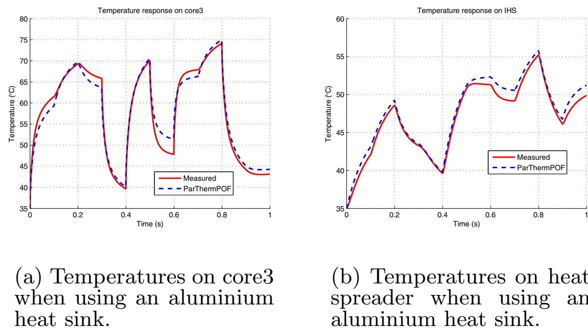
6. CONCLUSION



(a) Temperatures on cache when using a copper heat sink.

(b) Temperatures on core1 when using an aluminium heat sink.

Figure 13: Thermal simulation results on specific values of parameters



(a) Temperatures on core3 when using an aluminium heat sink.

(b) Temperatures on heat spreader when using an aluminium heat sink.

Figure 14: Thermal simulation results on specific values of parameters

Table 1: Errors of the peaks

Parameter settings	Maximal peak		
	Measured ($^{\circ}C$)	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%

In this paper, we have proposed a new parameterized thermal behavioral modeling method. The new method, *ParThermPOF*, builds the parameterized transient thermal behavioral models from the measured/given architecture thermal and power information. It could include a number of parameters such as locations of thermal sensors in a heat sink, different components (heat sink, heat spread, core, cache, etc.), thermal conductivity of heat sink materials, etc. *ParThermPOF* is a general top-down, black-box parameterized performance modeling techniques. It is very suitable for thermal-related design space exploration and optimization where both transient behavior and system parameters need to be considered. Experimental results on a practical quad-core microprocessor have showed that generated parameterized thermal models match the given power-thermal data very well.

Table 2: Average errors and relative errors between the computed and measured 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%

7. REFERENCES

[1] Intel multi-core processors (white paper), 2006. http://www.intel.com/multi-core.

[2] International technology roadmap for semiconductors(itrs) 2005, 2006 update, 2006. http://public.itrs.net.

[3] Multi-core processors - the next evolution in computing (white paper), 2006. http://multicore.amd.com.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proc. of Intl. Symp. on High-Performance Comp. Architecture*, pages 171–182, 2001.

[5] S. Gunther, F. Binns, D. Carmean, and J. Hall. Managing the impact of increasing microprocessor power consumption. In *Intel Technology Journal*, First Quarter 2001.

[6] Y. Hua and T. Sarkar. Generalized pencil of function method for extracting poles of an em system from its transient responses. *IEEE Trans. on Antennas and Propagation*, 37(2):229–234, Feb. 1989.

[7] Y. Hua and T. Sarkar. On svd for estimating generalized eigenvalues of singular matrix pencils in noise. *IEEE Trans. on Signal Processing*, 39(4):892–900, Apr. 1991.

[8] W. Huang, M. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proc. Design Automation Conf. (DAC)*, pages 878–883, 2004.

[9] D. Li, S. X.-D. Tan, and M. Tirumala. Architecture-level thermal behavioral characterization for multi-core microprocessors. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, 2008.

[10] Y. Li, B. C. Lee, D. Brooks, Z. Hu, and K. Skadron. CMP design space exploration subject to physical constraints. *Proc. IEEE Int. Symp. on High Performance Computer Architecture (HPCA)*, pages 15–26, 2006.

[11] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley-Interscience, 2002.

[12] M. Pedram and S. Nazarian. Thermal modeling, analysis and management in vlsi circuits: principles and methods. *Proc. of IEEE, Special Issue on Thermal Analysis of ULSI*, 94(8):1487–1501, 2006.

[13] T. Sarkar, F. Hu, Y. Hua, and M. Wick. A real-time signal processing technique for approximating a function by a sum of complex exponentials utilizing the matrix pencil approach. *Digital Signal Processing - A Review Journal*, 4(2):127–140, Apr. 1994.

[14] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature aware microarchitecture. In *Proc. IEEE International Symposium on Computer Architecture (ISCA)*, pages 2–13, 2003.

[15] J. M. Wang, B. Srinivas, D. Ma, C. C.-P. Chen, and J. Li. System-level power and thermal modeling and analysis by orthogonal polynomial based response surface approach (OPRS). *Proc. IEEE/ACM International conference on Computer-aided design (ICCAD)*, pages 728–735, 2005.

[16] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan. Efficient method for functional unit power estimation in modern microprocessors. In *Proc. Design Automation Conf. (DAC)*, pages 554–557, June 06.