General Parameterized Thermal Modeling for Multi-core Microprocessor Design

Thom Eguia, Sheldon X.-D. Tan, Ruijing Shen, Duo Li, Eduardo H. Pacheco, Murli Tirumala, Lingli Wang

Abstract—This paper proposes a new parameterized dynamic thermal modeling algorithm for emerging thermal-aware design and optimization for multi-core microprocessor design at architecture and package levels. Compared with existing behavioral thermal modeling algorithms, the proposed method can build the compact models from more general transient power and temperature waveforms used as training data. Such an approach can make the modeling process much easier and less restrictive than before, and more amenable for practical measured data. The new method, called ParThermSID, consists of two steps. First, the response surface method based on second 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 subspace system identification method, called ThermSID, is employed to build the discrete state space models, by construction of the Hankel matrix and state space realization, for each time-varying coefficient of the polynomials generated in the first step. To overcome the overfitting problems of the subspace method, the new method employs an overfitting mitigation technique to improve model accuracy and predictive ability. Experimental results on a practical quad-core microprocessor show that the generated parameterized thermal model matches the given data very well. The results also show that ThermSID is more accurate than the existing ThermPOF method. The compact models generated by ParThermSID also offer two orders of magnitude speedup over the commercial thermal analysis tool FloTHERM on the given example. ParThermSID is also much more general and flexible than the recently proposed parameterized thermal modeling method ParThermPOF.

I. INTRODUCTION

As VLSI technology is scaled into the nanometer region, the power density of high-performance microprocessors increases drastically. The exponential power density increase will, in turn, lead to average chip temperature to raise rapidly [3]. Higher temperature has significant adverse impacts on chip packaging cost, performance, and reliability. Excessive on-chip temperature leads to slower transistor speed owing to reduced carrier mobility, more leakage power consumption as leakage currents grow exponentially with temperature, higher interconnect resistance, and reduced reliability [11], [5].

One way to mitigate the high temperature problem is to put multiple cores into one single multi-core CPU [17], [1], [2]. In this way, one can simply increase the total throughput via task-level parallel computation, and have lower voltage and frequency to meet thermal constraints. In this case, however, the thermal effects are influenced by the placement of cores and shared caches. Therefore, it is very important to consider the temperature during the floor planning and architecture design of multi-core microprocessor.

The estimated temperature at the architecture level is vital for performing accurate power (especially leakage power), performance, reliability, wear-out and aging analysis in the floor planning and packaging design [24]. As a result, design guided by temperature can be optimized theoretically without potential thermal problems. For the cycle-accurate architecture thermal simulation, the simulation time can be very long (several seconds) [18], [27]. For instance, for a 3GHz CPU, 10K clock cycles (typically used) is 3.3us. For 10 seconds, the number of time steps is 3 million. Although the simulation techniques have seen some progress recently [6], more efficient thermal simulators are still highly desired. To facilitate this temperature-aware architecture design, it is important to have accurate and fast thermal estimation at the architecture level. The demands for reliable and practical tools for thermal architecture modeling from both architecture and CAD tool communities could not be higher.

The traditional bottom-up approaches including FEM (finite element), FDM (finite difference), and computational flow dynamics (CFD) based methods were widely used for thermal modeling and analysis in the past. They can be accurate when detailed thermal structures are known. However, these detailed models can be substantially large which prevents their use in many practical problems. Static and transient thermal modeling methods at different levels (parts, package, board) have been been proposed in the past. Many approaches try to use thermal resistance and capacitance with fixed topology networks subject to different thermal boundary conditions [14], [7], [4]. The main limitation of those methods is to determine appropriate RC values of elements, especially for complex geometries and boundary conditions. The RC values are typically determined and optimized against the field numerical or analytic results [10], [21] and measured data [23].

For thermal modeling at architecture level, existing work on HotSpot [12], [24] tries to solve this problem by generating the architecture thermal model in a bottom-up way based on the internal structure/architecture of the microprocessor. These bottom-up compact models, however, may suffer from...
accuracy loss, and compact models have to be calibrated with hardware if more accurate models are required. Also, the generated lumped RC models are not parameterized as different RC models will be generated with different parameters such as thermal conductivities, different thermal conditions (ambient temperatures), and packaging configurations [25], [26]. Recently a top-down behavioral architecture level thermal modeling method, ThermPOF, has been proposed [16], where temperature impulse responses are used to build the thermal models by the matrix pencil method.

In this paper, we propose a new parameterized thermal modeling approach for fast temperature estimation at the architecture and package levels for multi-core microprocessors. The new approach can build the behavioral thermal models from measured or simulated transient thermal and power information. The main advantage of the proposed modeling method over the existing black-box thermal modeling methods like ThermPOF [16] and ParThermPOF [15], where only impulse/step power inputs can be accepted is that the proposed method can accept general transient power and temperature waveforms. This makes the new method much more training friendly and general, as transfer function-like responses are typically difficult, even impossible (intractable), to obtain from the measurements. Furthermore, the new method is a top-down, black-box approach, which means it does not require any internal structure of the system. Lastly, it can accommodate a number of parameters, such as location of thermal sensors on a heat sink, thermal conductivity of heat sink materials, etc.

The new method, called ParThermSID, consists of two techniques. First, the response surface method (RSM) based on second order polynomials is adopted to build the parameterized models at each time point for all the given sampling nodes in the parameter space (except for time). Second it applies an improved subspace system identification method, called ThermSID to build the transient model for each time-varying coefficient of the polynomials generated in the first step. The subspace system identification method can accept general transient inputs and thus eliminates the need for impulse/step power inputs. The subspace system identification first generates the states of the desired models in terms of a Hankel matrix of Markov parameters from the measured input and output data via subspace projection and reduction. Then, the discrete state matrices are obtained through state matrix realizations. To overcome the over-fitting problem in the subspace method, ThermSID applies an over-fitting mitigation technique to pick up the best model among several that are built on partial training data to overcome the unavoidable overfitting problem associated with training-based modeling processes.

Experimental results on a real multicore microprocessor show that ThermSID and ParThermSID can provide thermal behavioral models that match the measured data very closely with similar accuracy to ThermPOF and ParThermPOF. The compact models generated by ParThermSID also offer two orders of magnitude speedup over the commercial thermal analysis tool FloTHERM on given examples. ParThermSID is also much more general and flexible than the recently proposed parameterized thermal modeling method ParThermPOF.

The rest of this paper is organized as follows: Section II presents the thermal modeling problem we are trying to solve. Section III reviews the subspace system method for its use in thermal modeling, while III-B explores overfitting and the resulting mitigation techniques. Section IV reviews the response surface method, and describes its use in parameterization. Finally, section V presents the results of both ThermSID and ParThermSID, with section VI concluding the paper.

II. PACKAGE-LEVEL PARAMETERIZED THERMAL MODELING PROBLEM

Our modeling problem requires building parameterized thermal models considering both time and other variable parameters of multi-core processors. The goal is to build a behavioral model with power and temperature as its inputs and outputs. The behavioral model must be dependent not only on power inputs, but on different system parameters as well. Two types of parameters are considered in our modeling problems. The first is time, with the remaining parameters to be discussed below.

We validate the proposed thermal modeling method by specifically looking at a quad-core microprocessor architecture from our industry partner, Intel Corporation. The architecture of this multi-core microprocessor, which contains four CPU cores (die 0 to die 3) and one cache core (die 4), is shown in Fig. 1. The temperatures are measured on the center of each die’s bottom face.

Fig. 2 shows a 3-D structure of this quad-core microprocessor, where the CPU die (with quad-cores) is located at the
bottom, and is in contact with the integrated heat spreader (IHS) in blue. At the top is the heater sink (HS), which has both top and bottom parts. The thermocouples (thermal sensors) in red are used to measure the real temperatures at specific locations, such as 5mm, 15mm, and 25mm away from the center of the heat sink. Typically, temperatures reduce as we move away from the center and bottom parts of the chip. Temperatures are hottest at the center of the core.

We also consider the thermal conductivity of the heat sink material as another parameter. Normally, a heat sink is made of either copper (Cu) or aluminum (Al). Cu and Al have different thermal conductivities, with Cu at 390W/(m·K) and Al at 240W/(m·K). Different heat sink materials may induce different temperature distributions on the chip. Fig. 3 shows the temperature distributions on the entire chip using a copper and an aluminum heat sink. The two figures show that the maximum temperature of the chip on a copper heat sink is 10°C less than that of the aluminum. The price of a copper heat sink, however, is apparently much higher than aluminum, so designers are met with 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. Such parameterized thermal models may prove helpful for design exploration and optimization.

Finally, this quad-core processor can be abstracted into a linear system with five inputs, eight outputs, and two parameters as shown in Fig. 4. The inputs are the power traces \( p_0 \) to \( p_4 \) of all the cores, and the outputs are the temperature responses on core0 to core3, cache, die, heat spreader, and heat sink for given parameters, which are represented by \( t_0 \) to \( t_4 \) in the figure. The parameters can be the location of the thermal sensors (distance to a center point), thermal conductivity of the heat sink material, etc.

![Fig. 4. Abstracted system](image)

Instead of using transfer functions to model the systems [16], which require step/impulse power inputs, we propose using a discrete linear time invariant system with state space models

\[
\begin{align*}
x_{k+1} &= Ax_k + Bu_k \\
y_k &= Cx_k + Du_k,
\end{align*}
\]

where \( A \in \mathbb{R}^{n \times n} \) is a stable matrix, \( B \in \mathbb{R}^{n \times m} \), \( C \in \mathbb{R}^{l \times n} \), and \( D \in \mathbb{R}^{l \times m} \). The input vectors \( u_k \in \mathbb{R}^{m \times 1} \) and output vectors \( y_k \in \mathbb{R}^{l \times 1} \) are the measured power input traces and temperature responses, respectively. \( x_k \) are estimated state variables (called Kalman states).

Given input \( u_k \) and output \( y_k \), the problem at hand is how to generate state matrices \( A, B, C, \) and \( D \), where \( D \) is typically considered a matrix of zeros. Note that given the discrete system matrices, the continuous system matrices can be obtained using methods such as Zero-Order-Hold, Impulse Invariance and Tustin Approximation [9].

III. SUBSPACE BASED TRANSIENT THERMAL MODELING – THERMSID

This chapter introduces the transient thermal modeling algorithm – ThermSID. We first start with the introduction of the subspace identification method.

A. Subspace system method

To determine the state matrices of a core’s thermal system, we apply a recently proposed subspace system identification method, called Numerical Algorithm for Subspace State Space System Identification (N4SID), to build the thermal models [20].

Subspace method basically builds high order prediction models by least-squares algorithm based on the transient inputs and outputs. The method differs from the previous method in that it builds (implicitly) the state variables (Kalman states) first, then realize the state matrices. The beauty of this method is that it is very numerically robust as only two numerical stable operations, QR (or its variant LQ) decomposition and singular value decomposition (SVD), are required and no iterative methods involved.

The whole algorithm flow is shown in Fig. 5. The algorithm consists of three major steps: (1) computing the Kalman states (implicitly) or, equivalently, state matrices (Hankel matrix) from the input and output data \( Y_{future}, U_{past}, Y_{past} \). Numerically, this is done by the least squares minimization method (via LQ decomposition); (2) performing the SVD on the combined coefficient matrices \( [ L_1, L_2 ] \) to obtain \( M_C \) and \( M_O \), which are the extended controllability and observability matrices; (3) realizing the system matrices \( [ A, B, C ] \) from \( M_C \) and \( M_O \) by least squares based regression. The detailed description of the algorithm is in the Appendix section (Section VII).

Compared to a previously proposed thermal modeling algorithm, ThermPOF [16], thermal modeling using the subspace identification method is more flexible and general. Since step responses are used during the training process in ThermPOF, a change to log-scale is necessary to better capture step response behavior. On the contrary, random power input traces and the resulting temperature responses of the desired system can be...
Data Matrix $\Phi = [Y^T_{\text{past}} \quad U^T_{\text{past}} \quad Y^T_{\text{future}} \quad 1^T]$

Hankel Matrix $H = [L_1 \quad L_2]$ 

Reduced SVD $H = U \Sigma V^*$

$M_0 = U \Sigma_1^{1/2}$, $M_c = \Sigma_1^{1/2} V_1^*$

Retrieve system matrices $A$, $B$ and $C$ from Extended Observability matrix and extended Controlibility Matrix

Fig. 5. The brief sketch of the subspace state-space system identification used directly as the input and output data sets for ThermSID, from which the system models are generated. Hence, it does not require the step temperature response as in ThermPOF. It also avoids log-scale change for time during the training process.

B. The overfitting problem

Simply applying the subspace-based method, however, can lead to overfitting problems during the modeling process. Overfitting occurs when a model describes random error or noise instead of the underlying behavior of the system, thus leading to poor predictive performance on new data sets.

Fig. 6. Training and verification waveform comparison, model order = 5

Typically, overfitting occurs when a model is built to follow its training data so closely that it harms the model’s predictive capabilities. The same phenomena occurs when using the subspace system method to generate the model. Fig. 6 provides an example of this using a model of order 10. We can see that the model provides a near exact match of the training data used to generate it. However, simulating a new set of inputs shows that the model is unable to follow the trend of the verification data.

Fig. 7. Verification waveform comparison, model order = 5, 10, 20

C. The proposed mitigating scheme

Improving the model’s predictive capabilities requires minimizing the complexity of the model itself. Fig. 7 shows the simulation results of three models of different order that are generated using the same training set. The model at order 20 exhibits the worst fit, but improves progressively as the model order is reduced. At order 5, the fit of the model has significantly improved. Furthermore, our tests indicate that order 5 is the optimal setting for our model, with no improvement gains to be made at lower orders. Lowering the complexity of the model, however, adversely affects the fit of the training data. Fig. 8 shows this result, where the quality of the fit decreases with the order of the model. The significance of this result will be explained later.

Despite lowering the complexity, the subspace system method is still unable to provide a reasonably accurate model. This result suggests that overfitting is still occurring and may be explored further. Several methods have been developed in the statistics community to mitigate the overfitting problem, which include early stopping, regularization, cross-validation, and Bayesian priors [13], [22]. For our problems, we observe that some sections of training data are more misleading than other sections due to highly correlated data that mask the underlying behavior of the system.

To show the effects of these correlated data, we shall
observe the differences in accuracy of a model created by the training data presented in Fig. 9. First, we shall create model 1 using the entire data set, and then generate model 2 with the dotted red section removed. Fig. 10 shows the simulation results of Core 0 on a separate verification set. The dotted red line is the waveform created by model 1, while the dashed green line is created by model 2. The results show a distinct improvement in accuracy when a particular section of data is removed. This misleading section of data appears to be detrimental to the predictive capabilities of the model. In this particular section, a step input value is present in each of the four cores.

Based on such observations, we design a specific overfitting mitigation scheme in which we iteratively remove some portions of the training data to search for better models. An obvious solution may be to remove all sections where there are simultaneous step inputs. However, doing this will remove most of the data, leaving us with an unusable training set. Furthermore, not all simultaneous step input occurrences are necessarily misleading. Removing certain sections indiscriminately may, in fact, harm the accuracy of the generated model. Therefore, a method of detecting and removing these misleading sections of data proves necessary.

To detect misleading sections of data, we employ a methodology that requires both a training set and validation set. This could be done by splitting an existing data set in half, with one portion used for training and the other for validation. Such partitioning scheme is similar to the Early Stopping method [13]. As described in Fig. 11, the training set is further partitioned into K sections. In each iteration, a section of data is omitted, with the rest used as training data. A modeling error $\epsilon$ is computed in each iteration by comparing the model created using the training data $i$ with the validation data set. In addition, the model is also compared to the entire training data set, resulting in a second modeling error value. The motivation for this comparison stems from the prior observation that the training waveform accuracy is lowered with low ordered models. It is further observed that removing sections of misleading data results in the improvement of the training waveform accuracy. Therefore, verification on two separate data sets provides a much greater insight into the predictive ability of the model. The final modeling error $\epsilon_i$ is thus calculated as the average of the errors for validation sets 1 and 2. Afterwards, the improved model can be chosen as the one with the lowest error value.

Fig. 12 presents the algorithm flow of the proposed thermal behavioral modeling method using the subspace system identification approach. Depending on the desired performance of the training data, the overfitting mitigation technique can be performed prior to utilizing the algorithm.

### IV. Parameterized Thermal Modeling Method – ParThermSID

To incorporate the parameter variables for the models, we apply the response surface method, which uses polynomials to represent the temperature responses with respect to the parameters. Then, the coefficients of the polynomials become functions of the time and power inputs, which are similar to the temperature responses for the non-parameterized problem and thus can be solved by ThermSID.

In summary, in order to build the parameterized behavioral model, we need to solve the following two problems: (1) finding response polynomial functions that can approximate the measured temperatures for all the controllable variables (parameters) with sufficient accuracy; (2) finding the state space models for polynomial coefficients (of the polynomials from step (1)) to capture the behavior of the temperature. For
problem (1), we introduce the response surface method to capture linear or nonlinear relationships between the parameters and responses (temperatures) at each time point. For problem (2), we apply ThermSID in the previous section to generate the dynamic models.

A. Response surface model for thermal response

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 [19]. This performance measure or quality characteristic is called the response, while the input variables are sometimes called independent variables.

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

\[
y = f(\xi_1, \xi_2, \ldots, \xi_k) + \varepsilon
\]

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

We must first transform the natural variables \( \xi \) in the range \([a, b]\) into coded variables \( z \) in the range \([-1, 1]\) using the linear transformation in (3).

\[
z_i = \frac{\xi_i - (b + a)/2}{(b - a)/2}
\]

After coding, the variable matrix \( Z \), defined in (7), will have all orthogonal columns, which can reduce numerical errors and increase numerical stability.

At time \( t \), a second-order response \( y \) depending on variables \( (z_1, z_2, \ldots, z_k) \) can be written as

\[
y = \beta_0(t) + \sum_{j=1}^{k} \beta_j(t)z_j + \sum_{j<i}^{k} \beta_{ij}(t)z_i z_j + \sum_{j=1}^{k} \beta_{jj}(t)z_j^2 + \varepsilon
\]

Notice that \( \beta_j(t) = \beta_j(t, p_0, p_1, \ldots, p_n) \) is a vector function of time and input powers. If we let \( z_{k+1} = z_1z_2, z_{k+2} = z_2z_3, \ldots, z_{k(k+1)/2} = z_2^{k+1} \), then (4) becomes

\[
y = \beta_0(t) + \sum_{j=1}^{q} \beta_j(t)z_j + \varepsilon
\]

which is a generalized linear regression model for coefficients \( (\beta_0(t), \beta_1(t), \ldots, \beta_q(t)) \), 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 (5).

Suppose that we have \( n \) observed responses \( y = (y_1, y_2, \ldots, y_n) \) and for each \( y_i \) we have one set of parameter values \( z_i = (z_{i1}, z_{i2}, \ldots, z_{iq}) \). (5) can then be written in matrix notation as

\[
y = Z\tilde{\beta}(t) + \varepsilon
\]

where

\[
y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, Z = \begin{bmatrix} 1 & z_{11} & z_{12} & \ldots & z_{1q} \\ 1 & z_{21} & z_{22} & \ldots & z_{2q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & z_{n1} & z_{n2} & \ldots & z_{nq} \end{bmatrix}, \tilde{\beta}(t) = \begin{bmatrix} \beta_0(t) \\ \beta_1(t) \\ \vdots \\ \beta_q(t) \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}
\]

We can minimize the squares of errors using the least squares estimator \( \tilde{\beta}_{est}(t) \) in (8).

\[
\tilde{\beta}_{est}(t) = (Z'Z)^{-1}Z'y
\]

In practice, we perform QR decomposition on \( X = QR \), where \( Q \) is orthogonal matrix and \( R \) is an upper triangular matrix, to make the computation more numerically stable, thus obtaining \( R\tilde{\beta}(t) = Q^T y \). After solving the linear equations, we get the estimated coefficient vector \( \tilde{\beta}_{est}(t) \).

B. Parameterized thermal behavioral models

After we obtain the coded variable matrix \( X \), the coefficients of our model can be computed using (8). At this point, we obtain the parameterized thermal model only on a single time point. We then compute the response surface models on all time points, which generates a set of response surface models, or more precisely, a set of coefficients, \( \tilde{\beta}(t) \). Each \( \tilde{\beta}(t) \) is a multi-input and multi-output function of time.

\[
\tilde{T}_{coeff}(t) = \tilde{\beta}(t, p_0, p_1, \ldots, p_n)
\]

In our specific case, \( n = 4 \) and \( \tilde{T}_{coeff}(t) \in R^5 \) is a vector for the temperature coefficients at time \( t \). Fig. 13 shows the
training coefficient responses (from 0 to 1 second) of $\vec{\beta}_0$ and $\vec{\beta}_1$, which are both functions of time. 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.

![Fig. 13. Coefficient responses using training data as power inputs](image)

(a) $\vec{\beta}_0$ coefficient response
(b) $\vec{\beta}_1$ coefficient response

After we obtain the coefficients $\vec{\beta}_i$, which are function
time and input powers, we can generate their system matrices $[A_i, B_i, C_i]$ for each coefficient $\vec{\beta}_i$ using ThermSID. Each
coefficient, a special temperature, and the input powers, will
become a multiple output and multiple input system based on
our thermal models.

**C. ParThermSID algorithm flow**

Fig. 14 summarizes the algorithm flow ParThermSID. In

**Algorithm: ParThermSID**

Input: temperature response vectors $y_i$ for each respective parameter, where $i = 1...l$, parameters $p_j$, where $j = 1...k$

Output: coefficient models for each $\beta_i(t)$, $i = 0...q$

1) Code parameter values $z_j$.
2) Form second order response equation in matrix notation as shown in (5) for each time step $t$.
3) Calculate $\vec{\beta}_{ext}(t)$ using (8) for each time step $t$.
4) Apply ThermSID to generate models for each coefficient $\beta_i(t)$ subsystem.

![Fig. 14. Parameterized thermal modeling algorithm – ParThermSID](image)

this algorithm, the first three steps compute the coefficients
of polynomials of parameters using response surface method.
Step 4 performs ThermSID to construct the dynamic coefficient
subsystems for each coefficient.

During the model evaluation, we first compute the responses
from all the coefficient systems. After obtaining the coefficient
responses, we can easily compute the temperature responses
using (5).

Compared to ParThermPOF in [15], ParThermSID can handle multiple input multiple output thermal systems. Furthermore, it can work for general power inputs and does not rely on the step power input training data, which is necessary for ParThermPOF.

**V. EXPERIMENTAL RESULTS**

This section verifies the effectiveness of the proposed
thermal modeling methods. The proposed method using the
subspace system identification method and overfitting tech-
nique is named ThermSID, while the parameterized version
is known as ParThermSID. We perform our tests on a quad-core
microprocessor architecture shown in Fig. 1 from our industry
partner Intel Corp. Intel built the detailed quad-core process-
model using FloTHERM [8]. The detailed model follows
the actual quad-core processors. The experimental
results were obtained from a Window PC workstation with
Intel Core 2 CPU (2.4Ghz, 2GB memory).

Both training and verification data sets were also provided by
Intel. All algorithms were implemented in Matlab 7.6

**A. Results for ThermSID with overfitting mitigation**

In this experiment, we shall compare the performance
of ThermSID with and without the overfitting mitigation
technique. We generate a model with ThermSID using the
entire training data set, and then generate a second model by
employing the overfitting mitigation scheme. We then compare
the accuracy of both models using a separate verification data
set. The training data set excites each core with step power
inputs that range between 0 to 20 Watts for a duration of 0 to
1 seconds. The verification data also excites the system with
step inputs, but only with values of 0 or 20 Watts. Fig. 15
and Fig. 16 show the training and verification power traces,
respectively. We partition the training data set into 20 sections
during the overfitting mitigation process.

![Fig. 15. Random power input traces for cores 0 to 3 and cache (set 1)](image)

We verify our algorithms by comparing their results with
the temperature response data of the verification set. Fig. 17
and Fig. 18 show the temperature response waveforms on Core
2 and the cache. The dotted line (Sid) represents the simula-
tion results generated by ThermSID without overfitting, while
the dashed line (ThermSID) represents the results generated
by ThermSID with overfitting. Lastly, the black line is the
measured temperature response from the verification data. The
figures clearly show the improvements in accuracy when the
overfitting mitigation process is employed in generating the
models.
Table I provides the percentage error statistics of cores 0 to 3 and the cache. The mean error and standard deviation are calculated for both Sid and ThermSID models. Results show that ThermSID significantly reduces the mean and standard deviation of errors in all regions. The most notable improvement occurs in the cache, where the mean error drops from 21.17 percent to 2.21 percent.

We then compare the performance of ThermSID against a previously proposed thermal modeling method ThermPOF.

Table III presents the error statistics of both methods. Note that since ThermPOF utilizes step responses to generate its model, we use a training data set with power inputs similar to that of Fig. 16 and their corresponding temperature response outputs. The models are then verified on a separate data set. With distinctly smaller mean error and standard deviation values, our results suggest that ThermSID exhibits better accuracy than ThermPOF.

We further verify the overfitting mitigation method by applying it to a second training data set with greater predictive ability. As with the first test, the training data set is divided into 20 sections, and Sid is used to generate the model. Fig. 19 compares the temperature response of Core 1, from which we can see that the model created by ThermSID exhibits an improvement in accuracy. Table II shows that the improvements carry over to all regions of the core, with a decrease in both mean error and standard deviation. The results of our tests on training set 2 show that the overfitting mitigation scheme is still applicable even on data sets that exhibit a fair amount of accuracy.

Our experimental results show that utilizing the overfitting mitigation method in conjunction with the subspace based modeling method significantly increases the accuracy of its generated model. Such an optimization approach is useful...
when sets of training data are scarce, and the present data is unable to generate an acceptably predictive model. While more optimal methods of model verification exist, the overfitting mitigation approach enables the creation of an improved model under such restrictive circumstances.

B. Result for ParThermSID

This section verifies the effectiveness of the proposed parameterized ParThermSID. An optimal training data set is used to generate the sub thermal systems mentioned in Subsection IV-B, which are of order \( m = 25 \). The accuracy of the generated model is then measured using two verification sets. A comparison of error statistics between ParThermSID and ParThermPOF [15] is also provided.

![Fig. 19. Temperature response of the verification data, Sid, and ThermSID for Core 1](image)

(a) Temperature response waveforms at core 0 and 3 using verification set 2. The graphs clearly show that the models follow the original waveforms very closely for both aluminum and copper heat sink materials.

![Fig. 20. Power input traces in Watts](image)

(a) Power input traces for training and verification sets 1 and 2

![Fig. 21. Temperature response waveforms considering Aluminum and Copper heat sink materials](image)

(b) Temperature response waveforms at 0mm and 5mm

![Fig. 22. Temperature response waveforms of the heat sink at different distances while considering aluminum and copper materials](image)

(a) Temperature response waveforms at 0mm

(b) Temperature response waveforms at 5mm

The power inputs of both training and verification data, as shown in Fig. 20, excites each core with a series of step-like power inputs of 0 or 20 Watts for a duration of 0 to 1 second. Notice that more than one power inputs can be active in this case, which is different from the more restrictive requirement of the one-input-active power inputs in ParThermPOF [15]. A sample is provided at every 0.001 second, which gives a data set of approximately one thousand samples.

We verify the proposed method by comparing its results with the temperature responses of verification sets 1 and 2. Fig. 21 provides the temperature response waveforms on cores

![Table IV](image)

<table>
<thead>
<tr>
<th>core</th>
<th>Aluminum</th>
<th>Copper</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std</td>
</tr>
<tr>
<td>Core 0</td>
<td>0.4</td>
<td>0.48</td>
</tr>
<tr>
<td>Core 1</td>
<td>0.48</td>
<td>0.43</td>
</tr>
<tr>
<td>Core 2</td>
<td>0.62</td>
<td>0.49</td>
</tr>
<tr>
<td>Core 3</td>
<td>0.8</td>
<td>0.55</td>
</tr>
<tr>
<td>Cache</td>
<td>0.82</td>
<td>0.57</td>
</tr>
<tr>
<td>hs(0mm)</td>
<td>0.29</td>
<td>0.57</td>
</tr>
<tr>
<td>hs(5mm)</td>
<td>0.35</td>
<td>0.29</td>
</tr>
<tr>
<td>hs(15mm)</td>
<td>0.39</td>
<td>0.23</td>
</tr>
<tr>
<td>hs(25mm)</td>
<td>0.49</td>
<td>0.27</td>
</tr>
</tbody>
</table>

All values are given as percentages.
TABLE V
STATISTICS OF ERROR PERCENTAGES AND MAXIMUM PEAK ERRORS FOR VERIFICATION SET 2. BOTH ALUMINUM AND COPPER ARE CONSIDERED. ALL VALUES ARE GIVEN AS PERCENTAGES.

<table>
<thead>
<tr>
<th></th>
<th>Aluminum</th>
<th>Copper</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>core 0</td>
<td>0.64</td>
</tr>
<tr>
<td></td>
<td>core 1</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td>core 2</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>core 3</td>
<td>0.58</td>
</tr>
<tr>
<td></td>
<td>cache</td>
<td>0.58</td>
</tr>
<tr>
<td></td>
<td>hs(0mm)</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>hs(5mm)</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>hs(15mm)</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td>hs(25mm)</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Tables IV and V provide the error statistics and maximum peak errors of verification sets 1 and 2. The mean error and standard deviation, as well as the maximum peak errors, are calculated for each core, the cache, and at distances of 0, 5, 15, and 25mm from the center of the heat sink. Both aluminum and copper heat sink materials are also considered. The tables show that the models generated by ParThermSID consistently exhibit less than 1 percent error in all three criteria.

In terms of CPU, ParThermSID is able to generate a model from training data in 5.52 seconds. The evaluation (1000 time steps) based on the parameterized model takes 0.48 seconds to finish. While ParThermPOF generates the similar model in 57.4 seconds and the evaluation time (also 1000 time steps) is 0.21 seconds. So ParThermSID is about 10X faster than ParThermPOF. The evaluation time of ParThermPOF is fast because the model is based on the partial fraction forms in terms of poles and zeros. By comparison, FloTHERM, which is a commercial analysis tool, performs modelling and simulation in about 34 minutes on a similar computing platform at Intel. As a result, we can see that ParThermSID is about two orders of magnitude faster than FloTHERM even though ParThermSID is implemented on Matlab.

TABLE VI
MEAN AND MAXIMUM PEAK ERROR COMPARISON BETWEEN ParThermSID AND ParThermPOF ON SEVERAL LOCATIONS CONSIDERING DIFFERENT PARAMETERS. ALL VALUES ARE GIVEN AS PERCENTAGES.

<table>
<thead>
<tr>
<th>loc/param</th>
<th>ParThermSID</th>
<th>ParThermPOF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean std maxpk</td>
<td>mean std maxpk</td>
</tr>
<tr>
<td>hs(ALU)</td>
<td>0.89 1.21 0.16 0.48</td>
<td>0.83 0.99 0.35</td>
</tr>
<tr>
<td>hs(ALU)</td>
<td>0.83 0.59 0.49 0.33</td>
<td>0.82 0.51 0.43</td>
</tr>
<tr>
<td>hs(ALU)</td>
<td>0.41 0.51 0.43 0.42</td>
<td>0.39 0.49 0.98</td>
</tr>
<tr>
<td>c0(Cu)</td>
<td>0.61 0.33 1.69 2.56</td>
<td>0.61 0.33 1.69 2.56</td>
</tr>
<tr>
<td>c1(Al)</td>
<td>0.52 0.03 1.28 0.28</td>
<td>0.52 0.03 1.28 0.28</td>
</tr>
<tr>
<td>c2(Al)</td>
<td>1.33 0.70 0.55 1.17</td>
<td>1.33 0.70 0.55 1.17</td>
</tr>
</tbody>
</table>

Lastly, Table VI provides a comparison between ParThermSID and ParThermPOF. Mean and maximum peak error comparisons are taken at specific parameters on the same quad-core microprocessor model using the same verification set. The second row, for example, which was taken at 5mm from the center of an aluminum heat sink, shows that both models have less than 1 percent error in both criteria. The results show that ParThermSID can provide models of similar accuracy to ParThermPOF. However, while ParThermPOF is limited to using step/impulse responses for its training method, ParThermSID is able to utilize general input and measured output data for training as shown earlier. Furthermore, in the parameterizations process, ParThermSID is able to handle multi-input/multi-output systems, which consolidates the coefficient calculations in both linear and second order equations.

VI. CONCLUSION
In this paper, we have proposed a new parameterized thermal behavioral modeling method. The new method, ThermSID, and its parameterized version, ParThermSID, build the parameterized dynamic thermal behavioral models from accurately computed thermal and power information. Compared to existing behavioral thermal modeling algorithms, the proposed method does not require step temperature responses for model training, which leads to greater flexibility during the modeling process. ParThermSID 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. 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 shown that the generated parameterized thermal models match the given power-thermal data very well. The compact models provided by ParThermSID offer two orders of magnitude speedup over the commercial thermal analysis tool FloTHERM on the given examples. ParThermSID is also more general and flexible than the recently proposed ParThermPOF method, which uses restrictive power inputs.

VII. APPENDIX
In this appendix section, we gives a brief introduction to the subspace identification method with some implementation details.

We suppose dynamic thermal system is linear time invariant (LTI) with noise as described in (1). For the system in (1), $h_k = CA^{k-1}B$ for $k \geq 1$ are called Markov parameters, which are the values of discrete-time impulse responses. The Hankel matrix of the Markov parameters can be defined as

$$H_{i,j} = \begin{bmatrix} h_i & h_{i+1} & \ldots & h_{i+j} \\ h_{i+1} & h_{i+2} & \ldots & h_{i+j+1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{i+j} & h_{i+j+1} & \ldots & h_{i+2j} \end{bmatrix}.$$  \hspace{1cm} (10)

It can easily shown that

$$H_{1:N-1} = M_O(N)M_C(N),$$  \hspace{1cm} (11)

where $M_C$ and $M_O$ are the extended controlability and observability matrices, which have the following form

$$M_C(j) = \begin{bmatrix} B & AB & \ldots & A^{j-1}B \end{bmatrix}, M_O(j) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{j-1} \end{bmatrix}.$$  \hspace{1cm} (12)
Considering $s$ points of history and $r$ points of future data, we first need to estimate the state variable $\hat{x}_k$ for the state $x_k$,

$$y_{\text{future}}(k) = M_O(r)\hat{x}_k + S_{\text{future}}u_{\text{future}}(k)$$  \hspace{1cm} (13)

where

$$y_{\text{future}}(k) = [y_k, y_{k+1}, \ldots, y_{k+r-1}]^T,$$  \hspace{1cm} (14)

$$u_{\text{future}}(k) = [u_k, u_{k+1}, \ldots, u_{k+r-1}]^T,$$  \hspace{1cm} (15)

$$S_{\text{future}} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ CB & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{-2}B & CA^{-3}B & \ldots & 0 \end{bmatrix}$$  \hspace{1cm} (16)

It can be shown that we can approximate $x_k$ in a least squares sense as

$$\hat{x}_k = K_1y_{\text{past}}(k-s) + K_2u_{\text{past}}(k-s) + K_3u_{\text{future}}(k),$$  \hspace{1cm} (17)

where

$$y_{\text{past}}(k-s) = [y_{k-s}, y_{k-s+1}, \ldots, y_{k-1}]^T,$$

$$u_{\text{past}}(k-s) = [u_{k-s}, u_{k-s+1}, \ldots, u_{k-1}]^T.$$  \hspace{1cm} (18)

From (13) and (17), and use $\hat{x}_k$ in (13) to estimate $\hat{y}_{\text{future}}(k)$, $\hat{y}_{\text{future}}(k)$ becomes a linear combination of $y_{\text{past}}(k-s)$, $u_{\text{past}}(k-s)$, and $u_{\text{future}}(k)$. Rewrite it in matrix form as follows,

$$\hat{Y}_{\text{future}} = L_1Y_{\text{past}} + L_2U_{\text{past}} + L_3U_{\text{future}},$$  \hspace{1cm} (19)

where

$$\hat{Y}_{\text{future}} = [\hat{Y}_{\text{future}}(s+1), \hat{Y}_{\text{future}}(s+2), \ldots, \hat{Y}_{\text{future}}(N-r+1)],$$

$$Y_{\text{past}} = [y_{\text{past}}(1), y_{\text{past}}(2), \ldots, y_{\text{past}}(N-r-s+1)],$$

$$U_{\text{past}} = [u_{\text{past}}(1), u_{\text{past}}(2), \ldots, u_{\text{past}}(N-r-s+1)],$$

$$U_{\text{future}} = [u_{\text{future}}(s+1), u_{\text{future}}(s+2), \ldots, u_{\text{future}}(N-r+1)].$$

We perform SVD on $H = [L_1, L_2] = USV^*$, where $S$ is used to identify the system order $n$. Afterwards, we can define the first $n$ columns of $U$ and $V$ as $U_1$ and $V_1$, respectively, and the diagonal matrix with the first $n$ singular values in $S$ as $\Sigma_1$. Then, we define

$$M_O = U_1\Sigma_1^{1/2}, \quad M_C = \Sigma_1^{1/2}V_1^*.$$  \hspace{1cm} (23)

The system matrix $C$ is the first $n$ rows of $M_O$, and $B$ can be retrieved from the first $n$ columns of $M_C$. The matrix $A$ can be obtained by

$$A = (M_O)^{-1}M_C,$$  \hspace{1cm} (24)

where $X^+$ means the pseudo inverse operation on $X$ as $M_O$ may not be a square matrix and invertible, and

$$M_O = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{M-1} \\ CA^{M-2} \end{bmatrix}, \quad M_C = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{M-2} \end{bmatrix}.$$  \hspace{1cm} (25)

REFERENCES


Summary of Revision (paper id: No. TVLSI-00130-2010)

VIII. REVIEW SUMMARY

First, we would like to thank the Editor in Chief and associate editor and the three referees for their careful reviews of our manuscript and suggestions and comments that help further improve the presentation of our work. A revision of the paper has been completed, all the review comments have been addressed and we hope the revised paper will resolve all the concerns and win the approval of the reviewers.

Before we reply the questions of each reviewer, we would like to highlight the following major modifications:

• Revised the subspace identification algorithm and moved the detailed description into the appendix section.
• Added more relevant references and comparison discussions.
• Added CPU times for the ParThermSID and ParThermPOF.
• All the mentioned presentation and language issues have been addressed.

Below, we will answer the questions raised by each reviewer. For the completeness, the original comments are quoted first, followed by a short explanation of how the comments have been addressed.

IX. ANSWER TO REVIEWERS’S COMMENTS

A. Answers to Reviewer #1’s Comments

Review Number 1.

This paper proposes a fitting algorithm to derive chip and package level compact thermal model for fast architecture-level thermal simulations during design time. I found this paper interesting but there are also many aspects needed to be clarified and/or improved for this work to be published in a journal. I also don’t think this paper is a good fit for TVLSI. It should fit better for TCAD, instead.

Detailed comments: - In the title, what does "macromodeling" mean? This is a vague term which doesn’t give the reader any information about the level this thermal model is applied to.

We changed "Macromodeling" to "modeling" in the title.

- Why does this work only fit to "multicore" design, as indicated in the title?

- The amount of mathematical definitions and listings is overwhelming without clear explanation, which makes it extremely hard for a reader to have patience/time to fully understand. It is the responsibility of the authors to explain the algorithm in a understandable way. After all, this is not a math paper. A simple example, in Equation (1), what do n, m, l mean? what is the vector x_k? Writing fancy equations without good explanation is not the right way for a good paper, IMHO.

The proposed method is based on the subspace identification method, which requires a brief explanation for completeness of the paper. At the same time, we tried to avoid spending too many pages on it as it was proposed before. We added some simple explanations and a illustrative flow of the algorithm in Fig. 5 to make the algorithm more easy to comprehend.

We rewrote the algorithm by summarizing the three major steps in the algorithm and moved the detailed description of the algorithm to the appendix (Section VII). The presentation has also been re-organized to make it more easy to follow.
- It is emphasized the model is parameterized. However, I had a hard time finding out how exactly is the "parameterization" carried out? For example, if I want to explore the difference between copper vs. aluminum heatsink, do I need to have separate training sets for both cases, or I just need to have one training set for one material, and the other material’s impact on the matrices can be derived with simple calculations? If it is the former case, the model is not parameterized.

- A related previous work is missed. It is by C M Lasance et al for a European project named PROFIT, which deals with extracting transient behavior thermal model from measurements for chips and packages.

We explained the parameterization in Section IV. The parameterization was done by separate training sets (the first method mentioned by the reviewer) as we applied the Response Surface method (RSM). We added more explanation at the beginning of the section to make this more clear.

We cited their works and other related works with some discussions in the introduction section. Thanks.

B. Answers to Reviewer #2’s Comments

Review Number 2.

This paper proposed a parameterized thermal modeling algorithm for CMP systems. The proposed algorithm first use second-order polynomials to build the parameterized models based on the response surface method. For each coefficient of the polynomial, which is also a multi-input multi-output function of time, the subspace system identification method is applied to build the discrete state models. To address the problem overfitting, the paper proposed using a refined training dataset (intentionally omitting part of the training data and pick the best model) for training purpose. Experimentally results indicate the proposed method is faster and more accurate than existing algorithms.

However, I have the following concerns:

1. Page 1, right column, para. 3, the paper argues that thermal models used by other thermal analysis tools such as ISAC are very time-consuming. The authors listed the runtime is typically several seconds. First of all, such work has been significantly improved afterwards. Please refer to: X. Chen, R. P. Dick, and L. Shang, Properties of and Improvements to Time-Domain Dynamic Thermal Analysis Algorithms, in Proc. Design, Automation, & Test in Europe, Mar. 2010. Second, the runtime reported in the paper indicates it takes 21.3 seconds to train the model and return simulation results. This seems to contradict the claim that several seconds is a long time for dynamic thermal analysis. I understand that the overhead of model training may be amortized over multiple runs. However, the runtime for model evaluation alone is not reported in the paper.
We added the evaluation the experimental results for *ParThermSID* (about 0.48s) and *ParThermPOF* (about 0.21s) and cited the mentioned paper.

2. Page 1, right column, para. 4, the argument that HotSpot cannot easily handle thermal conductivities, ambient temperatures, and packaging configurations is incorrect. My personal experience with HotSpot implies such parameter values can be easily specified in a configuration file. Please either justify your claim or fix it.

Here, we want stress that the generated lumped RC models in HotSpot are not parameterized as different RC models have to be generated with different parameter settings. In this paper, we try to incorporate those parameters into the models directly so that we do not need to rebuild the models with different settings (just like NMOS device models with channel width and length as variables). We revised the sentence to better reflect this.

3. Page 1, right column, para. 3, couldn’t -> could not

Fixed, thanks.

4. I like the idea that ParThermSID can handle parameters such as the locations of thermal sensors.

5. Page 2, left column, para. 4, The speed comparison with FloTHERM looks unnecessary to me. I’d rather see a speed comparison between ParThermPOF and ParThermSID. I understand ParThermPOF can only take step input response as input. However, I still would like to see the speed comparison between ParThermSID with ParThermPOF since the accuracies of the two methods are somewhat similar.

We have added the training times and evaluation times for both *ParThermSID* and *parThermPOF* in Subsection V-B. The training and evaluation times are different from the previous submission as we used a different workstation this time.

6. Page 3, Figure 4, what are p0 - p4 and t0 - t4?

7. Page 3, Equation 1, what is x? Also, below the equations, the dimensions such as "nxn" should be changed to n\times n.

8. Page 3, it would be nice to have a overview figure that illustrates how the ThermSID and the ParThermSID models fit together.

9. Page 3, Equation 3, please provide the corresponding reference.

10. Page 4, left column, "...It can be proven that [L1 L2] = H ...". What is H?

We added explanations for all mentioned notations in the paper. Thanks.
We created a new subsection (Subsection IV-C) for the flow of whole *ParThermSID* algorithm. We added more explanation in the section and revised Fig. 14 to better show how ThermSID and ParThermSID work together.
Note that the equation numbers changed in the revised version. For question 9, the proof of equation (11) is straightforward by multiplying $M_O(N)$ with $M_C(N)$ directly and no citation is needed.

We explained the $H$ (Hankel matrix) in the paper also.

11. Page 4, left column, Equation 16, what is the exponent on the right hand side of the equation?

12. Page 4, right column, para. 1, take out the last sentence since the system noise is never mentioned in the paper.

For (24), we explained the exponent $\dagger$ in the paper (which is the pseudo inverse operation). We changed the exponent to $+$ to be consistent with the notations used elsewhere.

We removed the last sentence for the noise discussion.

13. Page 7, left column, para. 3, stablem -> stable. Also, what is $Q$?

Typo has been fixed. $Q$ is obtained from QR decomposition of $X$. We added more explanation to make this more clear.

14. Page 7, right column, algorithm, I do not understand how you obtained the parameters $x_{-}(j)$, $j = 1$ to $k$.

15. Page 8, right column, last sentence, I do not understand it.

16. Page 10, Table IV, Table V, Table VI, why $\text{maxpk} < \text{mean}$ in a lot of cases? Shouldn’t $\text{maxpk} > \text{mean}$ in all cases?

We changed the notation for parameter $x_i$ into $z_i$ as $x_i$ has different meaning in the paper. The parameters are given by the users. For instance, the thermal conductivities of the heat sink.

C. Answers to Reviewer #3’s Comments

Review Number 3.

In this paper, the authors propose a new parameterized dynamic thermal modeling algorithm for emerging thermal-aware design and optimization for multi-core microprocessor design. While traditional approaches, such as HotSpot, are bottom-up approaches, this paper tries to do behavioral architecture level thermal modeling in a top-down manner, based on their previous work (ThermPOF, etc) for improved accuracy and less reliance on calibration with hardware. The authors then apply an over-fitting mitigation technique to overcome the over-fitting problem in their subspace method. Experimental results are then presented, based on a real multi-core microprocessor, to show that Thermsid and ParThermSID can provide thermal behavioral models that match the measured data very closely to ThermPOF and ParThermPOF.

This seems to be a good quality paper, but there are several concerns. First, it is not clear how the authors conduct their evaluation on a real multi-core microprocessor. It is not even mentioned what processor it is. Please explain the evaluation methodology in detail. Second, the methods proposed in this paper (ThermSID and ParThermSID) seem a little incremental compared to
the authors’ previous work, i.e., ThermPOF and ParThermPOF. The authors should make the contributions of their work clearer to the readers. Finally, the presentation of the paper needs some improvement. There are quite some typos in the paper. For example, page 1: modleing -> modeling. Please do more thorough proofreading. Also, the authors sometimes use Thermsid and sometimes use ThermSID. Please be consistent. Furthermore, Fig 2 is too dark to see clearly. Please fix.

For the first question, the experimental results were obtained using the thermal and power information from our industry partner Intel Corporation. Intel built the detailed quad-core process model using FloTHERM. The detailed model follows closely to the actual quad-core processors in Intel. Therefore, we can consider the experimental results were performed on practical industry multi-core processors.

The main difference between this work and the previous work (ThermPOF, and its variant, ParThermPOF) is that ThermSID and ParThermSID can take general power and thermal waveforms as inputs for the training process while ThermPOF can only take the step input with one input active at a time. Such power inputs are practically very restricted and difficult to obtained especially from measured data. For instance, one can’t easily turn off the caches once one of CPU cores is active in reality. This makes the proposed ThermSID and parThermSID more attractive from a practice point of view. We revised the paper to add those discussions.

We tried to fix as many typos and language issues as possible in the revised version.