Architecture-level Thermal Behavioral Characterization For Multi-Core Microprocessors *

Duo Li
Dept. of Electrical Engineering
University of California
Riverside, CA 92521
dli@ee.ucr.edu

Sheldon X.-D. Tan
Dept. of Electrical Engineering
University of California
Riverside, CA 92521
stan@ee.ucr.edu

Murli Tirumala
Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052
murli.tirumala@intel.com

ABSTRACT

In this paper, we investigate a new architecture-level thermal characterization problem from behavioral modeling perspective to address the emerging thermal related analysis and optimization problems for high-performance multi-core microprocessor design. We propose a new approach, called *ThermPOF*, to build the thermal behavioral models from the measured architecture thermal and power information. *ThermPOF* first builds the behavioral thermal model using generalized pencil-of-function (GPOF) method. And then to effectively model transient temperature changes, we proposed two new schemes to improve the GPOF. First we apply logarithmic-scale sampling instead of traditional linear sampling to better capture the temperature changing characteristics. Second, we modify the extracted thermal impulse response such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. To further reduce the model size, Krylov subspace based model order reduction is performed to reduce the order of the models in the state-space form. Experimental results on a practical quad-core microprocessor show that generated thermal behavioral models match the measured data very well.

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 [5, 4].

One way to mitigate the high temperature problem to put multiple CPUs or cores into one single chip [9, 1, 3]. In this way, one can simply increase the total throughput by parallel computation, and have lower voltage and frequency to meet thermal constraints. But the thermal effects are influenced by the placement of cores and caches. So it is very important to consider the temperature during the floorplanning and architecture design of multi-core microprocessor.

The estimated temperature at the architecture level can then be used to perform power, performance, and reliability analysis, together with floorplanning and packaging design [12]. As a result, design decision is guided by temperature and design is optimized theoretically without potential thermal problems. 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 community are currently lacking reliable and practical tools for thermal architecture modeling. Existing work on the HotSpot project [8, 12] 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 [14]. Also the resulting model work for only single CPU architecture and the accuracy may be not sufficient as many approximations are made.

In this paper, we propose a new thermal behavioral modeling approach for fast temperature estimation at the multi-core thermal architecture level at early design stage. The new approach, called *ThermPOF*, builds the transfer function matrix from the measured architecture level thermal and power information. It first builds behavioral thermal model using generalized pencil-of-function (GPOF) method [6, 7, 11], which was developed in the communication community to build the rational modeling from the measured data of real-time and electromagnetism systems. However, direct use of GPOF will not generate stable useful thermal models. Based on the characteristics of transient temperature behaviors, we make two new improvements over the traditional GPOF: First we apply logarithmic-scale sampling instead of traditional linear sampling to better capture the temperatures change over the time. Second, we modify the extracted thermal impulse response such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. Further we reduce the order of thermal models by Krylov subspace method for saving more during simulation process [13]. Experimental results on a practical multi-core microprocessor show that the generated thermal behavioral models can be built very efficiently and the resulting model match the measured temperature well.

The rest of this paper is organized as the follows: Section 2 presents thermal modeling problem we try to solve. Section 3 reviews a generalized pencil-of-function (GPOF) method for extracting the poles and residues from the transient response. Section 4 presents our new thermal behavioral modeling approach based on the GPOF. Section 5 explains a Krylov subspace method to reduce the model order for faster simulation. Section 6 shows the experimental results and Section 7 concludes this paper.

2. ARCHITECTURE-LEVEL THERMAL MODELING PROBLEM

We first present the new thermal behavioral modeling problem. Basically we want to build the behavioral model, which is excited by the power input and product the temperature outputs for the specific locations in the floorplanning of the multi-core microprocessor. Our behavioral models are created and calibrated with the measured temperature and power information from the real chips. The benefit of such behavioral thermal models is that it can easily built for many different architecture with different thermal conditions and thermal parameters (thermal conductivity, cooling configuration).

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

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

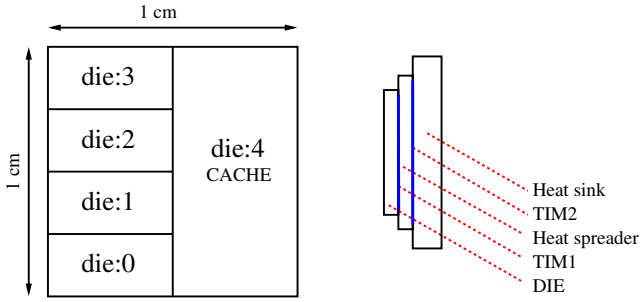


Figure 1: Quad-core architecture

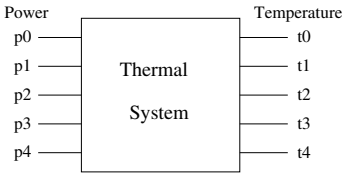


Figure 2: Abstracted system

Since the given the temperature data are transient and changing over time, we need to capture the transient behavior of the temperature, which can be achieved by building the impulse function between temperature and power in the time domain.

In this paper, we specifically look at quad-core microprocessor architecture from our industry partner to validate 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 temperature of each die is reported on the die bottom face in the center of each die. We can abstract this quad-core CPU into a linear system with 5 inputs and 5 outputs as shown in Fig. 2. The inputs are the power traces of all the cores, and the outputs are the temperature of them, respectively.

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

$$\mathbf{H(t)} = \begin{bmatrix} h_{00}(t) & h_{01}(t) & h_{02}(t) & h_{03}(t) & h_{04}(t) \\ h_{10}(t) & h_{11}(t) & h_{12}(t) & h_{13}(t) & h_{14}(t) \\ h_{20}(t) & h_{21}(t) & h_{22}(t) & h_{23}(t) & h_{24}(t) \\ h_{30}(t) & h_{31}(t) & h_{32}(t) & h_{33}(t) & h_{34}(t) \\ h_{40}(t) & h_{41}(t) & h_{42}(t) & h_{43}(t) & h_{44}(t) \end{bmatrix} \quad (1)$$

where h_{ij} are the impulse response function for output port i due to input port j.

Given a power input vector for each core $\mathbf{u}(t)$, the transient temperature can be then computed

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

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

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

where $\mathbf{y}(s)$, $\mathbf{u}(s)$ and $\mathbf{H}(s)$ are the Laplace transform of $\mathbf{y}(t)$, $\mathbf{u}(t)$ and $\mathbf{H}(t)$, respectively. $\mathbf{H}(s)$ is called the transfer-function matrix of the system where each $h_{ij}(s)$ can be represented as the partial fraction form or the pole-residue form (4).

$$h_{ij}(s) = \sum_{k=1}^{n} \frac{r_k}{s - p_k} \quad (4)$$

where $h_{ij}(s)$ is the transfer function between the jth input terminal and the ith output terminal; p_k and r_k are the kth pole and residue. Once transfer functions are computed, the transient responses can be easily computed.

The remaining important problem is to find the poles and residues for each transfer function h_{ij} from the measured thermal and power information. It turns out the generalized pencil-of-function can be used for this propose. But we cannot simply apply GPOF method as we show in the Section 4. In the following section, we will briefly review the GPOF method before we present our improvements.

3. REVIEW OF GENERALIZED PENCIL-OF-FUNCTION 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, 11]. It works on the sum of exponential forms, which can be represented 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 3, 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 $Y_2 = Z_1 R Z_0 Z_2$.

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

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

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

Figure 3: 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.

4. NEW ARCHITECTURE-LEVEL THERMAL BEHAVIORAL MODELING METHOD

In this section, we present our new thermal behavioral modeling approach based on the GPOF method mentioned in the previous section.

GPOF method is applied to the thermal impulse response, which in generally cannot be obtained directly from measurement. Instead, we measure the thermal step response for each core excited by the power input on the given multi-core microprocessor. Then impulse response can be obtained by performing the numerical differentiation on the step response.

But directly applying the GPOF to the computed thermal impulse response may not lead to stable and accurate model.

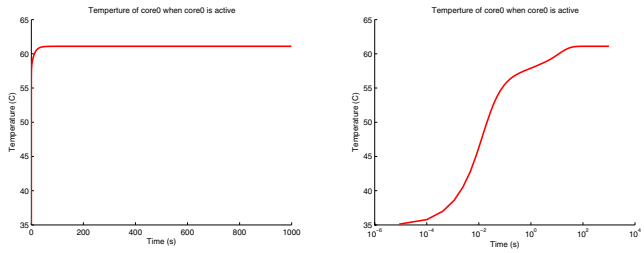
In the following, we will present two improvement schemes in *ThermPOF* method such that the resulting model is stable and accurate.

4.1 Logarithmic scale sampling for poles and residues extraction

The first problem we face for the thermal modeling is that linear sampling in the traditional GPOF method does not work for our thermal data.

According to GPOF method reviewed in Section 3, we know that matrices Y_1 and Y_2 are constructed from the sampled data and that the sampling time interval Δt must be the same. However, how to obtain sample data from the observed temperature curve became a big issue in our CPU temperature simulation, because the step temperature response often goes up drastically in the first few seconds and gradually tends to reach a steady state after a relatively long time.

This can be observed in Fig. 4(a), which is step temperature response for $core0$ ($die:0$) when only $core0$ is driven by a step $20W$ power source beginning at $t = 0$ (which is called *active* in this paper). The environment temperature, initial temperature when no input power at the beginning, is $35°C$. We observe that almost all the temperature increase occurs within the first second, from $35°C$ to $57.9°C$, where $61.1°C$ is the final temperature when $core0$ reaches a steady state after $1000s$ or more.



(a) Linear time scale. (b) Logarithmic time scale.

Figure 4: The transient temperature change of core0 when core0 is excited by 20W power input.

Hence we observe that temperature changes 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. Because to capture the thermal change information, sampling interval needs to very small, but this will lead to a very large number of sampled data due to the long tail of the thermal response reaching the steady state. As a result, we have to use a very big N and consequently very big L, which cause large dimensions of matrices Y_1 and Y_2 in GPOF. As a result, the following matrix operations such as multiplication, inverse or singular value decomposition (SVD) become very expensive.

In this paper, we propose to use logarithmic-scale sampling (log-scale sampling for short) to mitigate this problem. For the same temperature response in Fig. 4(a), we can obtain the log-sampled temperature response in Fig. 4(b), which clearly show how the temperature changes over the log-scale time gradually.

After the logarithmic operation of the time, the converted time, which is ln(time), will become negative. So we need to offset it to make sure that temperature response always starts at $t = 0$. And the offsetting will not affect GPOF operations. After we obtain the transfer function from GPOF, we need to compute the response in original time scale. We can get the response back by using (5),

$$\mathbf{y}'(t) = \mathbf{y}(ln(t) - ln(t_0)) \tag{5}$$

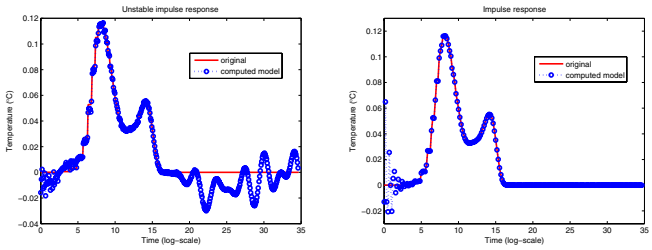
where $\mathbf{y}'(t)$ is the response in normal time scale $\mathbf{y}(t)$ is the response in log-scale; t_0 is the offset and usually it is a very small value.

4.2 Stable poles and residues extraction

4.2.1 Stable pole extraction

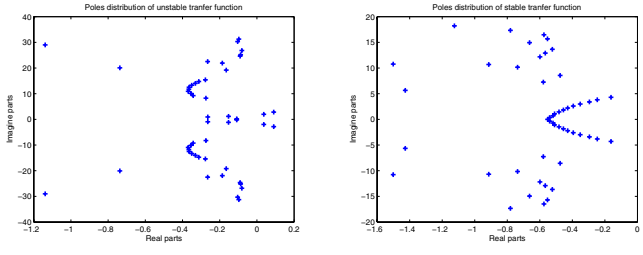
The second problem with the GPOF method is that it will not always generate stable poles for a given impulse response. Actually GPOF model can give a very good matching for a given impulse response for the sampled interval while using positive poles. But outside the sampled interval, the response from the model by GPOF can be unbounded due to the positive poles.

Fig. 5(a) shows the extracted impulse response compared to the original one for one of the cores. For this example, the sampled time interval is from 0 to 1000 seconds. Except for the very beginning (we will address this issue later), it can be seen that the computed model matches very well with the original one from time 0 to the $1000s$ (the corresponding $x = 18.55$ in log-scaled x-axis with offset being 8.8×10^-6). But outside the time interval, if we extend the time scale to 10^{10} seconds, they are significant difference between the two models. The computed models does not look like an impulse responses and will go unbounded actually owning to the positive poles. Fig. 6(a) shows the extracted poles where not all the poles extracted by GPOF are stable (having negative real parts).



(a) Impulse response with positive poles (b) Impulse response with only negative poles

Figure 5: Unstable and stable impulse response for Core0



(a) Existing positive poles (b) Only negative poles

Figure 6: Poles distributions of unstable and stable extracted transfer function

To mitigate this problem, we propose to extend the time interval for zero-response time. For any impulse response, after sufficient time, the response will become zero (or numerically become zero) as the area integration of the impulse

curve below is a constant. By sufficiently extending the time interval for zero-response time in an impulse response, we can make all the poles stable. The reason is that if we have positive poles, after sufficient long time, the response will always go non-zero and eventually become unbounded assuming all the poles are different numerically, which is always true practically. If we ensure the zero response for a sufficient long time, all the poles must be stable because response contributed only by those poles can decay to zero.

Using the same example, if we extend the time interval to 10^{10} seconds, which actually does not increase significantly in log-scale, all the extracted poles become stable. Fig. 6(b) shows the extracted poles by extending zero-response time to $10^{10}s$ where all the poles are stable (with negative real part) and Fig. 5(b) shows the extracted stable impulse response. For different problems, we may need to find such a sufficient time period. For all our problem, we find $10^{10}s$ seems such a good sufficient time for our example.

4.2.2 *Stabilizing the starting response*

Sometimes, the obtained impulse response may become zero numerically for a short period because temperature changes at the very beginning is very slow. And long zero-response time at beginning may cause the significant discrepancies as shown in Fig. 5(b).

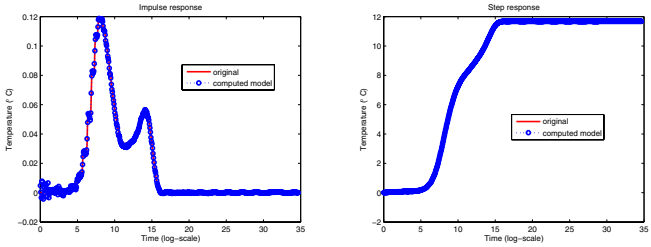


Figure 7: Impulse and step response with $L = 200$.

The reason for this problem is that the log-scaled impulse response is different from the typically impulse response from physical RLC circuit in which the response goes to non-zero immediately after $t = 0$. To resolve this problem, we may truncate the beginning zero-response time such that responses go to non-zero numerically immediately. Another way is by means of increasing the value of L, which means more sampling points but more accuracy. The advantage of the second method is we can set up the same offset for all the transfer functions, which can reduce the complexity in the thermal simulation we will discuss later. Fig 7 shows the improved impulse and step response from core0 to core1. Here $L = 200$, while $L = 100$ before. Notice that more L may not result in more accurate models. In our experiments, L varies from 150 to 300.

4.3 The ThermPOF modeling flow

Now, we describe all the important steps to obtain a transfer function of the thermal system, which is shown in Fig. 8.

To obtain the transfer function matrix of (1), we need to repeat the process for all the transfer functions required.

4.4 Recursive computation of temperature responses

After we obtain the transfer-function matrix **H** in Section 4, responses of the system can be computed theoretically whatever type of inputs are applied. In this paper, we introduce a fast recursive response computation method. By this method, we can fast compute the response in any time segment. The computation complexity is only $O(n)$, where n is the number of time segments or the number of power traces. Here, we assume the power inputs staying the same

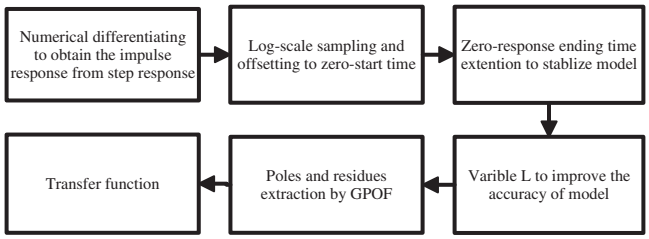


Figure 8: The flow of extracting one transfer function

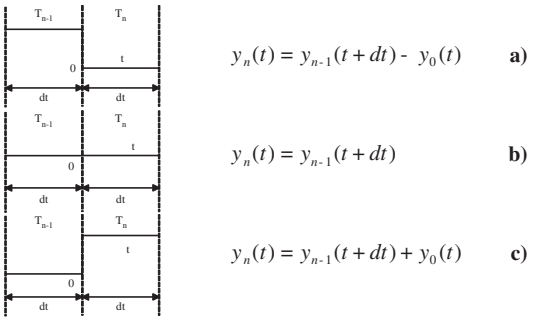


Figure 9: Recursive computation

during one time segment. So the power input can be seen as the sum of a group step inputs with different delays. This method works well for the general inputs as long as the time interval is small enough.

Given the power traces and time interval Δt, the response in each time segment not only depends on the current power inputs, but also depends on the previous power inputs. In total there are 3 cases of power inputs changes, as show in the left part of Fig. 9, where T_{n-1} and T_n are two immediate time segment, Δt is the time interval. Considering powers change at $t = 0$, we would like to compute the temperature $y(t)$ at t $(0 \le t \le \Delta t)$ in the nth time segment T_n. And $y(t)$ can be computed as shown in the right part of Fig. 9, respectively, where y_0 is the step response.

We remark that the proposed thermal modeling method is a general black-box modeling approach and can be easily applied to thermal modeling of microprocessors and other VLSI circuits at different granularities.

5. REDUCTION OF THERMAL MODELS

After we obtain our thermal models, we can further reduce the order of the models. In this paper, a classic Krylov-subspace model order reduction method PRIMA [10, 13] is used. Before using PRIMA, we need to have the state-space realization (6), which can be formed by poles and residues.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u \\ y &= \mathbf{c}^T\mathbf{x} \end{aligned} \tag{6}$$

In our model, poles and residues are both complex ande appear in conjugate pairs. And for each pair (7), the state-space realization is in the form of (8):

$$Y(s) = \frac{r}{s-p} + \frac{\overline{r}}{s-\overline{p}} \tag{7}$$

where $p = a + bj$ and $r = c + dj$.

$$\mathbf{A}_i = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \qquad \mathbf{b}_i = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \qquad \mathbf{c}_i^T = [c \quad d] \tag{8}$$

So we have the state-space realization of (9).

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{A}_1 & 0 & ... & 0 \\ 0 & \mathbf{A}_2 & ... & 0 \\ . & . & & . \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & \mathbf{A}_n \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ . \\ \vdots \\ \mathbf{b}_n \end{bmatrix} u \qquad (9)$$

$$y = \begin{bmatrix} \mathbf{c}_1^T & \mathbf{c}_2^T & ... & \mathbf{c}_n^T \end{bmatrix} \mathbf{x}$$

Then, the model order reduction is performed:

$$\mathbf{A}_r = \mathbf{V}^T \mathbf{A} \mathbf{V} \qquad \mathbf{b}_r = \mathbf{V}^T \mathbf{b} \qquad \mathbf{c}_r^T = \mathbf{c}^T \mathbf{V} \qquad (10)$$

where $\mathbf{V}$ is the projection matrix obtained from PRIMA.

After model reduction, we need to extract the poles and residues from the reduced matrix (10) to go back to the partial fraction form (pole-residue form). This can be done by means of the eigen-decomposition of $\mathbf{A}_r$, which leads to a diagonal matrix $\mathbf{\Lambda}$ containing eigenvalues and an orthogonal matrix $\mathbf{P}$ formed by the eigenvectors.

Thus, the transfer function of the pole-residue form can be computed using (11):

$$Y(s) = \sum_{k=1}^{q} \frac{\mu_k \cdot v_k}{s - \lambda_k} \qquad (11)$$

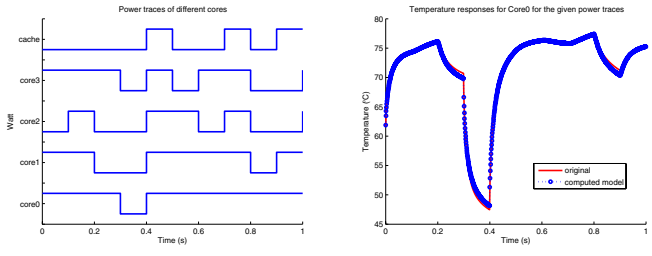
where λ_k is the kth diagonal element of $\mathbf{\Lambda}$, μ_k is the kth element of $\mathbf{c}_r^T \mathbf{P}$, v_k is the kth element of $\mathbf{P}^{-1} \mathbf{b}_r$ and q is the reduced order.

6. EXPERIMENTAL RESULTS

The proposed *ThermPOF* algorithm has been implemented in Matlab 7.0 and tested on the quad-core microprocessor architecture as shown in Fig. 1 from our industry partner. We first extracted transfer function matrix of the system through a training data set, which consists of the step responses for each core from other cores. After extracting of the transfer functions, we could apply them to obtain the thermal responses from other non-training power inputs with any time varying inputs.

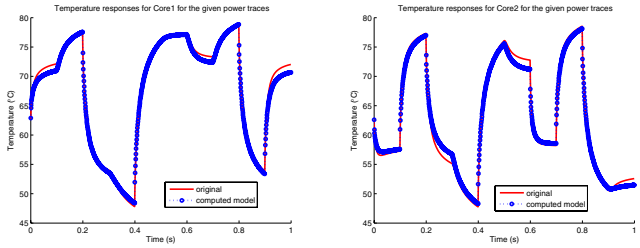
In our experiments, the temperature on every core is driven to an initial state by a power of $10W$ on each of the cores for 1000 seconds. Then we start to apply random power input traces as shown in Fig. 10(a), where the step power is $20W$ for all the cores.

We verify the correctness of our model from $0s$ to $1s$ based on the given thermal data from our industry partner. During that time, power inputs stay the same in each time step of $0.1s$. In practice, temperature on each core can be computed very fast by our model during any time interval, as the computation complexity in our model is only $O(n)$ by using recursive convolution, where n is the number of time steps.



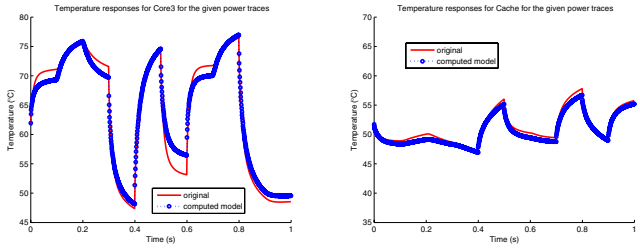
(a) Random power input traces. (b) Core0's temperature.

Figure 10: Thermal simulation results on given power input traces



(a) Core1's temperature. (b) Core2's temperature.

Figure 11: Thermal simulation results on given power input traces



(a) Core3's temperature. (b) Cache's temperature.

Figure 12: Thermal simulation results on given power input traces

Our simulation results for all cores are shown in Fig. 10(b), Fig. 11 and Fig. 12. The simulation runs very fast and costs only few minutes. From the figures, we can see that all the peak temperatures (both maximal and minimum) of each core during the whole time interval match well between the models and given measured date. The temperature errors are less than $1°C$, as shown in Table 2.

The errors (absolute values) between the original and our computed model are shown in Table 1, including the maximums, the means and the standard deviations. The maximal errors of *core*0, *core*1, *core*2 and *cache* are around $1°C$ or less than $2°C$ at the most. The maximal error of *core*3 is a little bit larger, which occurred during the time interval $0.5s - 0.6s$. But the standard deviations of the errors of all cores show that the temperature variations on average are less than $1°C$. Note that all the errors here are the absolute values of measured temperatures minus our computed temperatures.

Further we show the speedup gained in the simulation by the model reduction techniques presented in Section 5. In our experiments, we first set the order $M = 50$. After reduction, the order is reduced by $M = 30$ without loss of accuracy. All the errors are less than $< 2\%$ compared to the exact ones, as shown in Table 3. We know that simulation running time is positive proportional to the order of the models, especially for our analytical models represented by poles and residues. So from order 50 to 30, we can approximately reduce 40% simulation time, as shown in Table 4. We also set up order $M = 100$ at first, but the reduced order is still 30 if we want to maintain the high accuracy.

7. CONCLUSION

In this paper, we proposed a new architecture level thermal behavioral modeling method. The new method, *ThermPOF*, builds the thermal behavioral models by using transfer function matrix from the measured architecture thermal and

Table 1: Features of the errors between measured and computed temperatures ($M = 50$)

	Error ($^\circ C$)			Error percentage	
	Maximum	Mean	Std. deviation	Maximum	On average
Core0	1.05	0.34	0.23	1.56%	0.50%
Core1	1.67	0.53	0.48	2.44%	0.78%
Core2	1.78	0.61	0.47	2.56%	0.98%
Core3	3.33	1.10	0.82	6.09%	1.80%
Cache	1.05	0.63	0.22	1.84%	1.22%

Table 2: Errors of the maximal and minimum peaks ($M = 50$)

	Maximal peak			Minimum peak		
	Measured ($^\circ C$)	Error ($^\circ C$)	Percentage	Measured ($^\circ C$)	Error ($^\circ C$)	Percentage
Core0	77.27	0.45	0.58%	47.47	0.38	0.79%
Core1	78.86	0.04	0.05%	47.81	0.35	0.73%
Core2	78.55	0.38	0.48%	47.77	0.24	0.51%
Core3	76.48	0.75	0.98%	47.38	0.45	0.95%
Cache	57.80	0.99	1.72%	48.86	0.11	0.23%

Table 3: Errors of the maximal and minimum peaks and means ($M = 30$)

	Maximal peak		Minimum peak		Mean	
	Error ($^\circ C$)	Percentage	Error ($^\circ C$)	Percentage	Error ($^\circ C$)	Percentage
Core0	0.40	0.52%	0.46	0.96%	0.36	0.48%
Core1	0.12	0.15%	0.49	1.00%	0.47	0.69%
Core2	0.06	0.07%	0.34	0.70%	0.56	0.88%
Core3	0.76	0.98%	0.53	1.11%	1.11	1.66%
Cache	1.01	1.78%	0.01	0.02%	0.03	1.25%

Table 4: Speedup when $M = 30$ compared to $M = 50$

	Run time (s) when $M = 50$	Run time (s) when $M = 30$	Time reduced
Core0	1.31	0.80	38.9%
Core1	1.29	0.78	39.5%
Core2	1.28	0.78	39.1%
Core3	1.28	0.78	39.1%
Cache	1.30	0.79	39.2%

power information. We applied the generalized pencil-of-function (GPOF) method to construct the impulse response functions. To effectively model the transient temperature changes, we have proposed two new schemes to improve the GPOF. First we applied logarithmic-scale sampling instead of traditional linear sampling to better capture the temperature changing characteristics. Second, we modified the extracted thermal impulse response such that the extracted poles from GPOF are guaranteed to be stable without accuracy loss. We further reduce the generated thermal models by perform model order reduction using the Krylov subspace method, which leads to more saving in simulation of the reduced models. Experimental results on a practical quad-core microprocessor have demonstrated that generated thermal behavioral models match the measured data very well. The proposed method is a general black-box modeling approach and can be easily applied to thermal modeling of VLSI circuits at different granularities.

8. 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] 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.

[10] A. Odabasioglu, M. Celik, and L. Pileggi. PRIMA: Passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 645–654, 1998.

[11] 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.

[12] 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.

[13] S. X.-D. Tan and L. He. *Advanced Model Order Reduction Techniques in VLSI Design*. Cambridge University Press, 2007.

[14] 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.