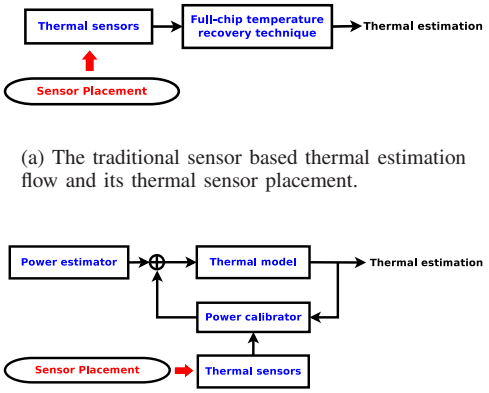
A Power-Driven Thermal Sensor Placement Algorithm for Dynamic Thermal Management

Hai Wang*, Sheldon X.-D. Tan†, Sahana Swarup†, and Xue-Xin Liu†
*School of Microelectronics & Solid-State Electronics,
University of Electronic Science & Technology of China, Chengdu, Sichuan, 610054 China
†Department of Electrical Engineering, University of California, Riverside, CA 92521 USA

***Abstract*—On-chip physical thermal sensors play a vital role for accurately estimating the full-chip thermal profile. How to place physical sensors such that both the number of thermal sensors and the temperature estimation errors are minimized becomes important for on-chip dynamic thermal management of today's high-performance microprocessors. In this paper, we present a new systematic thermal sensor placement algorithm. Different from the traditional thermal sensor placement algorithms where only the temperature information is explored, the new placement method takes advantage of functional unit power information by exploiting the correlation of power estimation errors among functional blocks. The new power-driven placement algorithm applies the correlation clustering algorithm to determine both the locations of sensors and the number of sensors automatically such that the temperature estimation errors can be minimized. Experimental results on a dual-core architecture show that the new thermal sensor placements yield more accurate full-chip temperature estimation compared to the uniform and the k-means based placement approaches.**

I. Introduction

Because of the exponential growth of the transistor density, excessive high temperature on chip has become the primary constraint in today's high-performance microprocessor design. For the modern multi/many-core architecture, dynamic thermal management (DTM) methods such as task migration, thread assignments, dynamic voltage and frequency scaling (DVFS) are used to regulate the temperature at runtime. However, for DTM to be effective, we need to know the full-chip thermal distribution profile instead of a few hot spots as both hot spots and thermal gradients need to be reduced to improve the performance and reliability of the chip. Specially, "cool spots" are also important besides hot spots since DTM methods such as task migration need both information to move the load from the hot spots area to the "cool spots" location. On-chip physical thermal sensors play a vital role for accurately estimating the full-chip thermal profile. However, since thermal sensors occupy many chip resources (areas, wiring and reading circuity), it is impractical to place a large number of thermal sensors across a chip. As a result, how to place the thermal sensors automatically to minimize both the number of sensors and thermal estimation error becomes an important topic.



(a) The traditional sensor based thermal estimation flow and its thermal sensor placement.

(b) The power calibration based thermal estimation with the new thermal sensor placement method.

Fig. 1. Comparison of the traditional sensor based thermal estimation flow and the power calibration based thermal estimation flow. The thermal sensors play completely different roles in the two approaches.

Thermal sensor placement problem for microprocessors has been studied in the past few years and several methods have been proposed [1], [2], [3], [4], [5], [6]. In [1], the maximum temperature difference from the hot spot to a certain spatial point on chip is shown analytically. In [2], a systematic thermal sensor placement method was proposed: an interpolation method was introduced to recover the full-chip thermal map, and the k-means clustering algorithm was applied to determine the sensor locations according to the hot spot distribution. Another thermal sensor placement work is the spatial thermal spectral-driven method [3]. By placing more thermal sensors at the places with high frequencies, the full-chip temperature can be recovered with higher accuracy. An optimization technique based thermal sensor placement method was introduced in [4]. In [5], the thermal correlation is exploited to assist the thermal sensor placement and thermal map recovering. And in [6], the authors proposed a sensor placement method by solving the on-chip hot spot tracking problem.

Although quite different in details, all the existing thermal

This research was supported in part by NSF grant under No. NSF-0902885 and No. 1255899, and Semiconductor Research Corporation grant under No. SRC 2009-TJ-1991 and an initial startup grant from UESTC.

978-3-9815370-0-0/DATE13/©2013 EDAA

sensor placement methods exploit only the chip thermal information and properties. As shown in Fig. 1 (a), full-chip thermal estimation is achieved by a full-chip temperature recovery technique with the thermal sensor readings as input. Power information, which is the source of the temperature, remains un-explored for thermal sensor placement. Power information is particularly relevant because power consumptions of many functional blocks are correlated and this can lead to less number of required sensors or better accuracy given the same number of sensors. In this paper, we propose a new thermal sensor placement method by looking into the information from the power consumption side. As shown in Fig. 1 (b), the proposed thermal sensor placement method is based on a different thermal estimation flow with two additional components: a performance counter based runtime power estimator and a thermal estimator with power calibration. The new thermal sensor placement method serves to boost the power calibration efficiency to achieve accurate thermal estimation.

II. Runtime thermal estimation framework

In this section, the runtime thermal estimation framework [7] is presented. We will first present the runtime power estimator which provides the input to the thermal estimator and at the same time introduces estimation error. Then, the error correction function of the thermal sensors in this framework is shown and how to place the thermal sensors to improve the thermal estimation accuracy is discussed.

A. Runtime power estimation

Due to the complex behavior of the microprocessor at runtime, accurately estimating the functional block (FB) level dynamic power is very hard. There are several off-line FB level power estimators available [8], [9]. They take the power event counts and multiply the counts by their corresponding unit powers to get the power estimations for FBs. These power estimators are considered to be accurate, for example, accurate to within 5% to 10% [10]. However, they are too expensive for runtime usage because there are too many power events to be monitored. Performance counter based runtime power estimators [11], [10] have much smaller overhead by monitoring only a few carefully chosen power events, and as a trade-off, has larger errors compared to the off-line power estimators. For example, in [11], at runtime, the power of the ith functional block is estimated as

$$U_i = E_i \times P_i \tag{1}$$

where E_i is the count of power event at the ith FB, P_i is the unit power per power event for the ith FB.

Although fast enough, the runtime power estimators are usually not accurate enough to be directly used for thermal estimation. Next, we will present how the power estimator errors can be compensated with the help of thermal sensors.

B. Runtime thermal estimation

The heat differential equation of the chip can be spatially discretized using finite difference method in the three dimensional space to generate an equivalent thermal circuit. Mathematically, for a chip with n_p functional blocks, if there are n discretized grids with specific boundary conditions, the equivalent thermal circuit can be modeled using an ordinary differential equation [12]

$$C\frac{dT(t)}{dt} + GT(t) = BU(t) \tag{2}$$

where $T(t) \in \mathbb{R}^n$ is the temperature vector containing the temperatures of the n thermal nodes, $C \in \mathbb{R}^{n\times n}$ is the thermal capacitance matrix, $G \in \mathbb{R}^{n\times n}$ is the thermal conductance matrix, $B \in \mathbb{R}^{n\times n_p}$ is the position matrix of the input where $B_{i,j}$ denotes the portion of the jth functional block power injects into the ith thermal node and $U(t) \in \mathbb{R}^{n_p}$ contains the power dissipations of the n_p functional blocks, whose estimations are shown in (1). The right hand side of (2) is also written as

$$J(t) = BU(t) \tag{3}$$

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

The accurate temperature T can be calculated from (2) using accurate power input. However, this is inapplicable because of the power estimation errors. Assume the power estimation from a runtime power estimator is $\bar{J}$, this will result in an *inaccurate* temperature estimation $\bar{T}$, which should be corrected to the actual value T.

In [7], an error compensation term ϵ

$$\epsilon \approx G\Delta T(t) \tag{4}$$

is introduced at $\bar{J}$ to get the calibrated power estimation

$$\tilde{J} = \bar{J} + \epsilon \tag{5}$$

which leads to a more accurate temperature estimation $\tilde{T}(t) \approx T(t)$, where $\Delta T(t) = T(t) - \bar{T}(t)$ denotes the thermal estimation error. This process is called power calibration as shown in Fig 1 (b).

However, only part of $T(t)$ and $\Delta T(t)$ is available where there are thermal sensors located. In order to calibrate the power with limited number of thermal sensors, the power error correlation among functional blocks needs to exploited.

First, assume there are n_s thermal sensors placed on chip. Please note that their placement may not be optimal without the thermal sensor placement algorithm proposed in this paper. For convenience, we first perform matrix permutation on (2) to group the thermal nodes with thermal sensors together as

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \frac{dT_s(t)}{dt} \\ \frac{dT_u(t)}{dt} \end{bmatrix} + \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} T_s(t) \\ T_u(t) \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} U(t) \tag{6}$$

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

Accordingly, (4) becomes

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} \Delta T_s(t) \\ \Delta T_u(t) \end{bmatrix} = \begin{bmatrix} \epsilon_s \\ \epsilon_u \end{bmatrix} \tag{7}$$

The value of ΔT_s is known, but ΔT_u is unknown due to the absence of thermal sensors. Since there are $2n - n_s$ unknowns

with n equations, (7) is unsolvable (in the normal sense) unless the number of unknowns is reduced. An *error correlation matrix* $D \in \mathbb{R}^{(n-n_s)\times n_s}$ is introduced by taking advantage of the power correlation among different functional blocks on chip, and ϵ_u is represented in terms of ϵ_s as

$$\epsilon_u = D\epsilon_s \tag{8}$$

which will reduce the number of unknowns in (7). Subsequently, the error compensation term ϵ can be calculated for accurate thermal estimation. All FBs related to the same thermal sensors form a new block, called *sensor block*.

Please note that (8) may not work well if the thermal sensor placement is not considered. If ϵ_u and ϵ_s are strongly correlated, then (8) will hold for all the time over a wide variety of applications. However, if ϵ_u and ϵ_s are weakly correlated, or even independent, there will be no D exist which can make (8) valid. Our objective in this paper is to find the optimal thermal sensor placement which will maximize the correlation between ϵ_u and ϵ_s and at the same time minimize the number of thermal sensors.

III. New thermal sensor placement algorithm

Our new thermal sensor placement algorithm mainly includes two steps: first, experiments, similar to [13], will be performed on a variety of benchmarks collecting the sample data to form a correlation graph. Then, a correlation clustering algorithm is applied on the correlation graph. The functional blocks are automatically clustered into sensor blocks without pre-specifying the sensor block count (number of thermal sensors). After the sensor placement, the correlation matrix D is determined.

A. Correlation graph generation

Please note that instead of finding the error relation for each thermal node as (8), it is only necessary to find the correlation among functional blocks since the powers of the nodes inside each functional block are extremely correlated. As a result, we only need to find the power error relation of functional blocks as

$$\Delta U_a = D_p \Delta U_s \tag{9}$$

where $\Delta U_a \in \mathbb{R}^{n_p}$ and $\Delta U_s \in \mathbb{R}^{n_s}$ represent the power error of each functional block and the power error of the functional blocks with thermal sensors, respectively, and $D_p \in \mathbb{R}^{n_p \times n_s}$ is the functional block level correlation matrix. The final D matrix can be calculated from D_p as shown at the end of this section.

As the first step, the correlation graph is generated for all the functional blocks, using the collected sample data, both from measurement and power estimator simulation.

Assume there are b benchmarks with steady power configurations. First, we run the benchmarks using the power estimator and record the power results

$$\hat{U} = [\hat{U}^1, \hat{U}^2, \ldots, \hat{U}^b] \tag{10}$$

where, for example, the ith sample

$$\hat{U}^i = [\hat{u}^i_1, \hat{u}^i_2, \ldots, \hat{u}^i_{n_p}]^T \tag{11}$$

since there are n_p functional blocks. Next, the benchmarks are run on the test chip until the temperatures reach steady state. The steady state temperature is measured as T. The real power of the chip is reversely calculated as

$$U = [U^1, U^2, \ldots, U^b] \tag{12}$$

using the measured temperatures as in [14], [15]. The errors of the functional block powers are obtained as

$$\Delta U = U - \hat{U} \tag{13}$$

The next step is to form a correlation matrix, such that functional blocks with high power error correlations can be identified and put into one sensor block. Using the data samples ΔU, the standard correlation matrix $corr_{\Delta u} \in \mathbb{R}^{n_p \times n_p}$ is calculated with the element at the ith row and jth column as

$$\frac{E[(\Delta u_i - \mu_i)(\Delta u_j - \mu_j)]}{\sigma_{\Delta u_i}\sigma_{\Delta u_j}} \tag{14}$$

where μ_i is the expected value of Δu_i.

By definition, the standard correlation matrix is a symmetric matrix containing the correlation values of each random variable pair. The correlation value is a number between -1 and 1 which reveals the dependence of a random variable pair, where 1 and -1 indicate the two random variables are fully dependent and 0 means totally independence. In our case, the absolute value of the correlation is taken.

The correlation graph is easily generated by observing the correlation matrix. Assume the correlation matrix of a chip with four functional blocks is

$$corr_{\Delta u} = \begin{bmatrix} 1 & 0.9 & 0.4 & 0.8 \\ 0.9 & 1 & 0.3 & 0.8 \\ 0.4 & 0.3 & 1 & 0.7 \\ 0.8 & 0.8 & 0.7 & 1 \end{bmatrix} \tag{15}$$

The corresponding correlation graph is generated as shown in Fig. 2.

B. Correlation clustering algorithm

The highly correlated functional blocks need to be clustered into the same sensor block in order to enhance the relation in (8). There are many clustering algorithms, such as k-means method used in [2] (although the objective is quit different). However, most of these clustering algorithms require the number of clusters (in our case, the number of sensors) to be known as *a priori*, which typically is not the case or estimation needs to be performed. For example, if there are four functional blocks, three of them are fully correlated and the other one is independent. There are clearly only two clusters, but k-means algorithm with $k = 3$ will lead to a non-optimal result with three clusters. As a result, it is better to determine the number of sensors in the clustering algorithm. In this paper, we introduce the correlation clustering algorithm [16] which can automatically determine the number of thermal sensors and their locations based on the generated correlation graph.

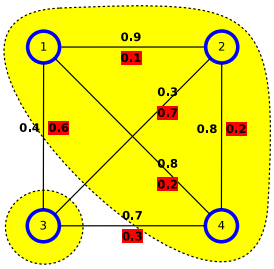


Fig. 2. A correlation clustering example for a complete undirected weighted graph with four vertices. On each edge e_{ij}, w_{ij}^- is shown as red shaded while w_{ij}^+ is in normal. The outputed clusters are surrounded by dashed lines filled with yellow color.

In our case, the input of the correlation clustering algorithm is the correlation graph generated previously, which can be expressed as a complete undirected weighted graph $\mathcal{G} = (V, E)$ (see Fig. 2 for example). There are n_p vertices, each represents a functional block. Let v_i denote the ith vertex, e_{ij} denote the edge (v_i, v_j) and the correlation on the edge is c_{ij}. There are two weights on each edge e_{ij}, denoted as w_{ij}^+ and w_{ij}^-, where w_{ij}^+ is the cost of cutting the edge and w_{ij}^- is the cost of keeping the edge. We use the additive weights which are calculated as $w_{ij}^+ = c_{ij}$ and $w_{ij}^- = 1 - c_{ij}$. The output of the correlation clustering algorithm is a new graph $\mathcal{G}'$ with edges $x_{ij} \in \{0, 1\}$, where $x_{ij} = 0$ means vertices v_i and v_j are assigned into the same cluster while $x_{ij} = 1$ means e_{ij} is cut and v_i and v_j are assigned into different clusters. According to the graph theory, the values of x_{ij} should also satisfy the triangle inequality: $x_{ij} + x_{jk} \geq x_{ik}$.

The basic idea of the correlation clustering algorithm is to find the optimal clusters (and the number of clusters) such that the uncorrelations inside each cluster is minimized and at the same time, the correlations among clusters are minimized. We can measure the uncorrelations inside clusters as

$$W_{inside} = \sum_{i,j:i<j} (1 - x_{ij}) w_{ij}^- \tag{16}$$

and the correlations among clusters as

$$W_{among} = \sum_{i,j:i<j} x_{ij} w_{ij}^+ \tag{17}$$

Please note $1 - x_{ij}$ is 1 if v_i and v_j are in the same cluster while 0 means they are separated. The total cost can be written as $W_{inside} + W_{among}$ and the formulation of the optimization problem is

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j:i<j} (1 - x_{ij}) w_{ij}^- + x_{ij} w_{ij}^+ \\ \text{subject to} \quad & x_{ij} \in \{0, 1\} \\ & x_{ij} + x_{jk} \geq x_{ik} \end{aligned} \tag{18}$$

As an example, consider the graph shown in Fig. 2. Obviously, v_1, v_2 and v_4 are highly correlated to each other and are assigned into the same cluster. For v_3, although it is relatively correlated to v_4, it is uncorrelated to v_1 and v_2. As a result, it is assigned into another cluster. These two clusters minimize the cost function in (18) and 2 is automatically determined as the number of clusters. On the contrary, for k-means based method, $k = 3$ for example, may lead to non-optimal results.

It should be noticed that sometimes the number of thermal sensors are limited due to the design constraint. On the other hand, it is also possible that the power errors among functional blocks are highly independent and as a result, the clustering correlation algorithm will generate a large number of clusters. In order to deal with these conflicts, the correlation clustering algorithm can be tuned by simply multiplying a scalar l to the standard correlation matrix $corr_{\Delta u}$, where $l > 1$ is used to reduce the number of clusters until the sensor number is smaller than the allowed number. In addition, $0 < l < 1$ can be used to increase the number of clusters in order to enhance the accuracy, if there are still space for more thermal sensors.

The clustering correlation algorithm is NP-hard but the complexity can be reduced by using heuristic or approximate methods [17]. Thermal sensor placement is an off-line algorithm, time complexity is generally not a concern. In addition, the number of functional blocks to be clustered is usually small (dozens or hundreds). In our experiment, the correlation clustering algorithm always finishes in a flash (within 1 second) and the memory cost is negligibly low.

C. Locate thermal sensors

We have clustered functional blocks into sensor blocks, then one thermal sensor has to be placed for each sensor block. We call the functional block with a thermal sensor located as the *sensor functional block*. Although several functional blocks are clustered into the same sensor block by power error correlations, their physical locations may be distributed all over the chip. We decide the sensor functional block as the one closest to the centroid of the sensor block. The thermal sensor is then put on the center of this functional sensor block. Please note that thermal sensors cannot always be put at the specified location because of the design considerations and limitations [4]. In this case, thermal sensor can be fine tuned within the sensor functional block. If the design constraint is not satisfied, the functional block which is the second closest to the sensor block centroid is used as the sensor functional block instead.

D. Error correlation matrix generation

In this subsection, we will present how to generate the error correlation matrix D. As introduced in III-A, we have to form D_p first, then generate D. We use the linear regression method to find the relations among the functional blocks within each sensor block. Assume the ith functional block is associated with the jth sensor functional block (which means they are clustered into the same sensor block and the jth functional block has a thermal sensor placed), the relation

$$\Delta u_j = a_j \Delta u_i \tag{19}$$

is found using the sample data information $[\Delta u_i^1, \Delta u_i^2, \ldots, \Delta u_i^b]$ and $[\Delta u_j^1, \Delta u_j^2, \ldots, \Delta u_j^b]$. Since the two functional blocks are clustered together, meaning they are highly correlated, the relation (19) will be statistically

good. With (19) for each functional block without thermal sensors, i.e. $j = 1, 2, \ldots, n - n_s$, D_p is populated with a_j.

The correlation matrix D is derived by multiplying a transformation matrix $M \in \mathbb{R}^{(n-n_s)\times n_p}$ as the following:

$$D = MD_p \tag{20}$$

The element at the ith row jth column of M is non-zero only if the ith node without sensor is inside the jth FB, and the non-zero value is

$$M_{i,j} = r_{u_i}/r_{s_i}, \tag{21}$$

where r_{u_i} is the power ratio of the ith node without sensor in the total power of its functional block (the jth functional block), and r_{s_i} is the power ratio of the corresponding sensor node in the total power of its functional block.

IV. Experimental results

The experiments are performed on a Linux server with 3.0GHz quad-core CPU and 16GB memory. A dual-core architecture shown in Fig. 3 (a) and (b) is used. As shown in Fig. 3 (c), a COMSOL [18] package model with the specified chip structure is built to calibrate our thermal model (2). The size of the chip is $10mm \times 10mm \times 0.7mm$. The simulation time step h is chosen to be $0.1s$ to balance the speed and accuracy. The ambient temperature is set to be $35°C$. We use wattch [9] to generate the power information with SPEC benchmarks [19]. The runtime power estimator is built by using two most important performance counts for each functional block power, where the parameter of each performance count is obtained through a linear regressor with samples from several benchmarks similar to [11], [10]. For the correlation clustering, we use the opensource software for [17] available at [20].

We first compare the new thermal sensor placement algorithm with the simple uniform placement method. Because in the new thermal sensor placement algorithm, the number of thermal sensors is determined automatically, we force the other placement methods to have the same number of thermal sensors for a fair comparison. In order to validate the new placement algorithm, the uniform thermal sensor placement has the same thermal estimation flow (shown in Fig. 1 (b)) as the new placement method. The correlation clustering algorithm inside the new thermal sensor placement method gives 6 clusters, which means there are totally 6 thermal sensors. For the uniform sensor placement, the thermal sensor is placed on L2 Cache, FPAdd, and IntExec, for each core. While for the new method, the sensors are placed on L2 Cache, FPReg, and FPAdd.

The error snapshot plots with the *bzip2* benchmark at 15s for the uniform and the new method are shown in Fig. 4 (a) and (b). Thanks to the runtime power estimator and the power calibrator, both the uniform and the new placement method generate quite good thermal map estimations with only 6 thermal sensors. However, the uniform thermal sensor placement does not consider the power error correlation at the sensor placement stage, such that the functional blocks inside each sensor block are not fully correlated, resulting in relatively large error at some positions. Since the new thermal sensor placement algorithm automatically groups the functional blocks with respective to power error correlations, the power calibrator works more efficiently in this case and generates the power map with much less error.

The new method is then compared against the existing k-means based thermal sensor placement method with interpolation thermal map recovery [2] on the same benchmark. The new thermal sensor placement method takes advantage of the runtime power estimator which is additional information compared to the k-means placement method. As shown in Fig. 4 (c), the k-means method has larger error compared to the new method because it is extremely hard (perhaps impossible) to accurately recover the full-chip temperature with *only* temperature information from 6 thermal sensors.

More accuracy comparison results on the other benchmarks are summarized in Table I. We have also increased the number of thermal sensors by multiplying the correlation matrix $corr_{\Delta u}$ with a constant term l as introduced in Section III-B, where the case of 10 thermal sensors is achieved by multiplying $l = 0.65$ and the case of 14 thermal sensors is reached by multiplying $l = 0.6$. We can see more thermal sensors actually improve the accuracy, but the number of sensors automatically determined by the new algorithm is already fairly enough.

V. Conclusion

In this paper, we propose a new systematic thermal sensor placement algorithm to optimize both the number of sensors and the thermal estimation errors. The new placement method explores the correlation of power estimation errors among functional blocks to determine the best locations of physical thermal sensors. The new algorithm applies the correlation clustering algorithm to determine both the locations of sensors and the number of sensors automatically in the optimal way in terms of temperature estimation errors. Experimental results show that the thermal sensor placements given by the proposed method lead to more accurate full-chip temperature estimation compared to the existing k-means based placement approach with interpolation based thermal estimation.

References

[1] K. Lee, K. Skadron, and W. Huang, "Analytical model for sensor placement on microprocessors," in *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, 2005, pp. 24–27.

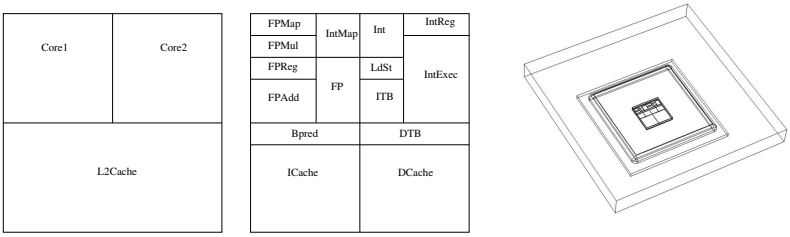
[2] S. Ogrenci Memik *et al.*, "Optimizing thermal sensor allocation for microprocessors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 516–527, March 2008.

[3] A. Nowroz, R. Cochran, and S. Reda, "Thermal monitoring of real processors: Techniques for sensor allocation and full characterization," in *Proc. Design Automation Conf. (DAC)*, 2010.

[4] S. Sharifi and T. S. Rosing, "Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 10, pp. 1586–1599, Oct. 2010.

[5] Y. Zhang, B. Shi, and A. Srivastava, "A statistical framework for designing on-chip thermal sensing infrastructure in nano-scale systems," in *Proc. Int. Symp. on Physical Design (ISPD)*, 2010, pp. 169–176.

[6] S. Reda, R. Cochran, and A. Nowroz, "Improved thermal tracking for processors using hard and soft sensor allocation techniques," *IEEE Trans. on Computers*, vol. 60, no. 6, pp. 841–851, June 2011.



(a) The dual-core microprocessor architecture.

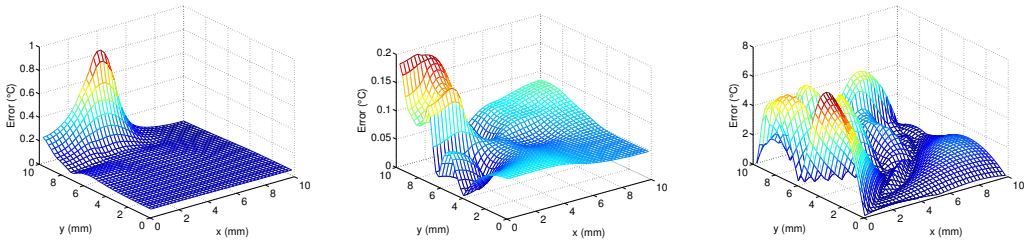
(b) The architecture for each core composed of FBs.

(c) The package model with specified architecture built using COMSOL.

Fig. 3. The dual-core microprocessor architecture, with two cores and one shared L2 cache, and the package model built using COMSOL.

TABLE I
ACCURACY COMPARISON OF THE NEW THERMAL SENSOR PLACEMENT ALGORITHM WITH THE UNIFORM AND THE K-MEANS THERMAL SENSOR PLACEMENT METHODS.

Bench	6 sensors						10 sensors						14 sensors					
	Uniform		New		k-means		Uniform		New		k-means		Uniform		New		k-means	
	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max	avg	max
bzip2	0.08	0.83	0.06	0.19	2.0	7.2	0.07	0.85	0.05	0.20	1.4	7.7	0.06	0.33	0.05	0.12	1.3	7.7
gzip	0.15	0.42	0.12	0.37	1.9	8.0	0.11	0.37	0.08	0.25	1.5	8.1	0.09	0.34	0.09	0.26	1.2	7.0
mcf	0.07	0.65	0.06	0.41	1.0	4.7	0.08	0.52	0.04	0.26	0.91	4.7	0.09	0.47	0.02	0.10	0.8	3.9
mgrid	0.04	0.66	0.04	0.22	2.3	10.4	0.04	0.48	0.03	0.16	1.7	10.4	0.03	0.31	0.02	0.12	1.5	7.6
swim	0.22	2.7	0.17	1.2	1.3	8.1	0.16	1.8	0.15	0.86	1.3	8.2	0.15	0.93	0.15	1.1	1.3	6.5
galgel	0.08	1.3	0.06	0.28	1.0	5.2	0.07	0.93	0.04	0.18	0.94	5.6	0.05	0.66	0.03	0.15	0.93	6.2



(a) Error plot of the uniform thermal sensor placement.

(b) Error plot of the new thermal sensor placement.

(c) Error plot of the k-means sensor placement with interpolation thermal map recovering.

Fig. 4. Error snapshot plot with the bzip2 benchmark at 15s. For both cases, 6 thermal sensors are placed on chip.

[7] H. Wang *et al.*, "Full-chip runtime error-tolerant thermal estimation and prediction for practical thermal management," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2011.

[8] S. Gunther *et al.*, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, vol. 5, February 2001.

[9] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. on Computer Architecture (ISCA)*, 2000, pp. 83–94.

[10] M. D. Powell *et al.*, "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters," in *Proc. IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)*, 2009, pp. 289–300.

[11] W. Wu *et al.*, "A systematic method for functional unit power estimation in microprocessors," in *Proc. Design Automation Conf. (DAC)*, June 2006, pp. 554–557.

[12] Y.-K. Cheng *et al.*, *Electrothermal Analysis of VLSI Systems*. Kluwer Academic Publishers, 2000.

[13] H. Wang *et al.*, "Runtime power estimator calibration for high-performance microprocessors," in *Proc. Design, Automation and Test In Europe. (DATE)*, March 2012, pp. 352–357.

[14] A. Nowroz, G. Woods, and S. Reda, "Improved post-silicon power modeling using AC lock-in techniques," in *Proc. Design Automation Conf. (DAC)*, 2011.

[15] R. Cochran, A. Nowroz, and S. Reda, "Post-silicon power characterization using thermal infrared emissions," in *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, 2010, pp. 331–336.

[16] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," *Machine Learning Journal*, vol. 56, no. 1-3, pp. 89–113, 2004.

[17] M. Elsner and W. Schudy, "Bounding and comparing methods for correlation clustering beyond ILP," in *Workshop on Integer Linear Programming for Natural Language Processing*, June 2009.

[18] www.comsol.com, "Comsol mutiphysics: User guide," *Version 4.1*.

[19] J. L. Henning, "SPEC CPU 2000: Measuring CPU performance in the new millennium," *IEEE computer*, vol. 1, no. 7, pp. 28–35, July 2000.

[20] "Correlation clustering system," http://www.cs.brown.edu/~melsner/.