Collaborative Calibration of On-Chip Thermal Sensors Using Performance Counters

Shiting (Justin) Lu, Russell Tessier, and Wayne Burleson
University of Massachusetts
Department of Electrical and Computer Engineering
Amherst, MA, USA
{jlu, tessier, burleson}@ecs.umass.edu

ABSTRACT

Thermal sensors are currently deployed in processors to collect thermal information for dynamic thermal management (DTM). The calibration cost for thermal sensors can be prohibitively high as the number of on-chip sensors increases. We propose an on-line multi-sensor calibration method which combines potentially inaccurate temperature values obtained from two sources: temperature readings from thermal sensors and temperature estimations using system performance counters. A data fusion strategy based on Bayesian inference, which combines information from these two sources, is demonstrated along with a temperature estimation approach using performance counters. The approaches are verified via simulation for an AMD Athlon 64 processor with 24 on-chip temperature sensors scaled to a 45nm technology node. Our results show that the standard deviation of temperature sensor measurement errors can be reduced from $3 \sim 4\ ^oC$ to $\leq 1\ ^oC$ using the proposed method. Additionally, our MATLAB implementation shows that the new approach runs at least 67x faster than competing approaches based on Kalman filtering making it highly appropriate for run-time use.

1. INTRODUCTION

Thermal management is a critical problem for modern processors due to high transistor density. This characteristic increases power consumption and heat density in a small silicon area causing performance degradation and decreased system reliability. As a result, dynamic thermal and power management strategies are often employed to tackle run time thermal and power issues [2][7]. On-chip thermal sensors deployed in processors are currently used to assist DTM and recent trends indicate increased future use to assess thermal gradients.

The efficiency and effectiveness of a DTM strategy relies on accurate thermal sensor temperature measurements. Two challenges exist in using these sensors: (1) detecting if a sensor is providing erroneous readings and (2) recalibrating the sensor, if necessary. Our focus in this work is the second challenge. Often, thermal sensor calibration involves performing thermal imaging using an infrared camera while capturing the physical readings of thermal sensors [6]. As the sensor count on a silicon die increases, the per-chip calibration cost can be prohibitively high, leading to on-chip thermal sensor use without individual sensor calibration. The temperature reading error of these *uncalibrated* thermal sensors can be substantial (up to 34 ^oC at 95 ^oC [13]) which adversely impacts DTM strategy. Even if thermal sensors are initially well-calibrated, their readings gradually drift away from actual temperature values due to device wear-out and recalibration is needed to regain the required accuracy. In general, it is not practical to perform in-field calibration with thermal imaging since end users typically do not have expensive calibration equipment.

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

The first contribution of this paper involves fast on-chip temperature estimation at run time using system performance counters. A linear model is built to estimate the local temperature at a specific location on the chip by including information gathered from system performance counters. In this paper, the term *estimated temperature* is exclusively used to refer to a temperature obtained from thermal estimation using performance counters. Experimental results from a collection of benchmarks show that our model can estimate the *relative* temperature differences across sensors with very good accuracy (a correlation coefficient between the actual and estimated thermal profile ≥ 0.9). Unlike other techniques that use performance counters to assist in temperature estimation [15][18], our approach *does not* require the use of power traces derived from performance counter information, simplifying our approach.

Our second contribution is a multi-sensor collaborative calibration algorithm (MSCCA) which combines estimated temperature profiles and sensor readings from multiple sensors to obtain more accurate temperature values for individual sensors. *Corrected temperature* values obtained from the MSCCA algorithm are used for thermal sensor calibration. The Bayesian inference technique integrated into MSCCA utilizes the implicit spatial correlation of estimated temperatures to correct sensor reading errors. An increased number of on-chip sensors provides increased MSCCA accuracy since more spatial correlation information is captured within the estimated temperatures.

To validate our algorithm, architectural and thermal simulators are used to collect performance counter and tem-

perature values, respectively. Results show that the mean error of thermal sensor readings can be effectively reduced to $\leq 1\ ^oC$ and the standard deviation of the errors in the corrected temperatures is reduced to an acceptable level (from $3 \sim 4\ ^oC$ to $\leq 1\ ^oC$). For comparison, the computational complexity and estimation accuracy of our new approach is evaluated against another method which is based on power estimation and Kalman filtering [18]. Our results show that the new calibration strategy has comparable accuracy but is more efficient (at least 67x faster).

This paper is organized as follows. Background and related work on sensor calibration and thermal estimation are discussed in Section 2. The thermal estimation scheme using performance counters is presented in Section 3. Section 4 describes the proposed algorithm for combining thermal estimation and sensor readings. Section 5 describes the experimental approach for verifying the effectiveness of thermal estimation and calibration. Experimental results are presented in Section 6. Section 7 concludes the paper.

2. BACKGROUND

On-chip thermal sensors determine temperature by measuring related physical quantities, e.g. frequency and voltage, and converting them to temperature values. The calibration of thermal sensors finds parameters for the mapping of physical quantities to temperature [12]. Three main calibration techniques exist. The first and most traditional way of calibrating on-chip thermal sensors measures the thermal profile of a running chip using thermal imaging technologies and reports the sensor readings at that time instant. The model parameters can then be obtained by solving a series of equations or by using statistical parameter inference. Usually, the calibration cost associated with this approach is very high since every chip experiences a different thermal imaging response and the amount of effort increases with the number of on-chip thermal sensors.

The second calibration technique uses design-for-calibration. This approach is implemented by integrating dedicated hardware circuits on chip which monitor the process variation around the thermal sensors [4]. With the knowledge of the chip process variation, the errors in thermal sensor readings can be compensated and model parameters can be optimized to reflect the physical relationship between the temperature and physical quantities. Since the process variation monitoring hardware consumes silicon real estate, it raises the chip cost when a large number of sensors are integrated.

A third calibration approach uses accurate on-chip thermal estimation instead of thermal imaging to determine actual temperatures. In Liu [9] and Cochran and Reda [3], the authors describe methods to construct spatial correlation models from measurement data which can then be used to recover the full chip temperature profile. Since the measurements are subject to noise, the specific amount of error at each sensor can be difficult to determine. As a result, most recent approaches for sensor calibration use a combination of sensor readings and other on-chip information to calibrate sensors.

Kumar, *et al.* [7] use the performance counters in an Intel Pentium-4 processor to estimate the overall chip temperature. For multiple sensor calibration, it is necessary to estimate the temperature at the micro-architectural level due to thermal gradients within the silicon die, so an estimation strategy with finer granularity is needed. Lee, *et al.* [8] proposed a run-time temperature sensing strategy using performance counters in high-performance processors. In this strategy, performance counters are used to estimate the power dissipation for each hardware component and the estimated power traces are then used to estimate the temperature trace based on the thermal model implemented in a thermal simulator. The mapping from power to temperature requires a complex thermal model which characterizes the thermal RC network of the given chip. In two recent papers [15][18], two sources of temperature information are combined: (1) noisy sensor readings and (2) localized power consumption which can be related to temperature. The technique used to integrate data from these two data sources is Kalman filtering (KF). Although power traces can be accurately estimated at run time [11], a thermal RC model is required to determine the mapping coefficients required to convert power dissipation to temperature in the prediction step of KF approaches. Unfortunately, the derivation of this model is not trivial due to the complexity of silicon materials. KF based approaches have shown the ability to track the temperature profile of a chip at a high computational cost since KF is performed each time a temperature estimation is made.

Like the other approaches mentioned in the previous paragraph, the calibration method developed for this paper fits into the third category of calibration approaches. Unlike previous techniques, we *directly* use information from performance counters for temperature estimation rather than using power consumption as an intermediate value for conversion between performance counter information and temperature.

Although the collaborative calibration of a number of sensors on a silicon die using performance information from multiple sensors is a new challenge, similar problems have been studied in the wireless sensor network community for years. For example, a Bayesian inference method was employed to reduce the noise of sensor data [5]. The approach combines a priori knowledge of the expected reading, the noise characteristics of the sensors, and an observed noisy reading to obtain a more accurate reading estimate. Whitehouse, *et al.* [16] formulated the calibration of a large sensor network into a parameter estimation problem. They determined that micro-calibration (the calibration of sensors one-by-one) is sometimes problematic due to the lack of a calibration interface and unobservable environments. Thus, the need for macro-calibrations (collaborative calibrations) which utilize the correlation among sensors arises.

3. THERMAL ESTIMATION USING PERFORMANCE COUNTERS

In a microprocessor, performance counters monitor run time system statistics, such as the load/store rate, branch prediction miss rate, amount of cache misses, and instructions per cycle (IPC), among others, for various system management purposes. Since these statistics contain the activity information of functional units in the processor, they can be used to estimate the power consumption at a per-structure granu-

Table 1: Correlation between integer scheduler temperature and performance counters for *velosity*

IPC	Branch Rate	Load Rate	Store Rate	L1I Miss Rate	L1D Miss Rate	ROB	FP
0.755	-0.721	-0.390	-0.279	-0.057	-0.610	0.279	-0.302
Instr.#	Iwin	Int Rate	Replay Rate	BPred Hit Rate	LD Forward	MemAcc	STQ
0.994	0.397	0.601	-0.382	0.198	0.925	-0.926	-0.668

larity using linear regression [11] or unit power consumption [8][17]. Unlike power consumption estimation, functional unit temperature estimation is more complex due to inter-component correlation resulting from heat flow across the chip. Consequently, the units surrounding an active unit could have relatively high temperature even though they are in inactive states. The area difference between functional units further complicates the problem because two units may have different temperatures even they have the same power consumption.

By virtue of these complexities, the temperature at a specific position on the chip is generally not linearly related to one particular performance counter, so it is not possible to construct a thermal map using a few independent performance counters. Table 1 shows the correlation coefficients between the temperature at the center of the integer scheduler and a subset of system performance counters determined by running the *velosity* benchmark on the SESC simulator [14]. Here, performance counters values and temperatures are generated from the architectural simulator SESC and thermal simulator HotSpot [1], respectively. From the table, it is apparent that most of coefficients are around 0.5 away from the values of 1 or -1, which represent a linear correlation. However, performance counters are correlated with each other. A sampling of counter pair correlations is shown in Table 2.

Table 2: Correlation between performance counters

IPC/ld	IPC/st	ld/st	FP/Int	DMiss/ld	ROB/IPC
-0.776	-0.305	0.561	-0.873	0.809	-0.691

The non-linear impact of performance counter correlation on temperature can be illustrated via a simple example (Note: the real relationships generally are more complex). Consider two performance counters, x and y, that are quadratically related, as shown in Equation (1).

$$y = \alpha_1 x + \alpha_2 x^2 \tag{1}$$

Also, the temperature, T, has a closed form representation based on these two variables given by the following equation.

$$T = \gamma_1 x + \gamma_2 x^2 + y \tag{2}$$

By replacing x^2 in Equation (2) with a reordered version of Equation (1), the following linear representation is obtained, where A and B are determined by Equation (4).

$$T = Ax + By \tag{3}$$

$$A = \gamma_1 - \alpha_1\gamma_2/\alpha_2 \quad \text{and} \quad B = 1 + \gamma_2/\alpha_2 \tag{4}$$

Effectively, since α and γ are constant, temperature can be approximated with a linear representation. Although this example is trivial compared to actual on-chip thermal analysis, it provides a basis for our model derivation in the next section assuming a sufficient amount of performance counters are available to be used to provide accuracy.

3.1 Linear Model for Temperature Estimation

A linear model can be built using values from system performance counters located in various locations in the processor to estimate the temperature of a specific unit in the processor, as shown in the following example for an integer scheduler (T^{IS}). In the following equation, M_{ij} is the value of performance counter j at time instance i. T_i^{IS} is the estimated temperature at time instance i using these counter values. The coefficients, β_j^{IS}, are determined in the model training phase which will be discussed in Section 3.3.

$$T_i^{IS} = \beta_0^{IS} + \beta_1^{IS} M_{i1} + \beta_2^{IS} M_{i2} ... + \beta_k^{IS} M_{in} \tag{5}$$

The above equation can be rewritten in matrix form, as shown in Equation (6), to represent thermal estimation for multiple locations. At time instance i, $\mathbf{T_i}$ is an $m \times 1$ column temperature vector and $\mathbf{M_i}$ is a $n \times 1$ column performance counter vector. $\boldsymbol{\beta}$ is an $m \times n$ coefficients matrix.

$$\mathbf{T_i} = \boldsymbol{\beta} \times \mathbf{M_i} \tag{6}$$

The estimated temperatures can be calculated quickly, in real time because only scalar multiplications and additions are involved in Equation (5).

3.2 Performance Counter Selection Criteria

The selection of performance counters is critical for achieving a good temperature estimation. Performance counters that give little correlation with temperature for most functional units are excluded from the estimation. Empirically, an independent relation is identified for a correlation coefficient of less than 0.2. One issue with the model that arises from throwing away irrelevant performance counters is model stability. A model that is suitable for some benchmarks does not work for other benchmarks, i.e. estimated temperatures differ greatly from the actual temperatures. This instability problem could be resolved by excluding some performance counters from the model although they may have good correlation with temperatures for some benchmarks. A reasonable rule of thumb is that performance counters with extremely large β_j coefficients make the model unstable.

The performance counter selection procedure involves a select-and-test iteration during the model training period, i.e. train the model using a set of selected performance counters and perform a cross-benchmark test (different benchmarks are used for training and testing) on the trained model.

3.3 Model Training

The model parameters in $\boldsymbol{\beta}$ are important factors which impact the model accuracy. As mentioned previously, $\boldsymbol{\beta}$ is

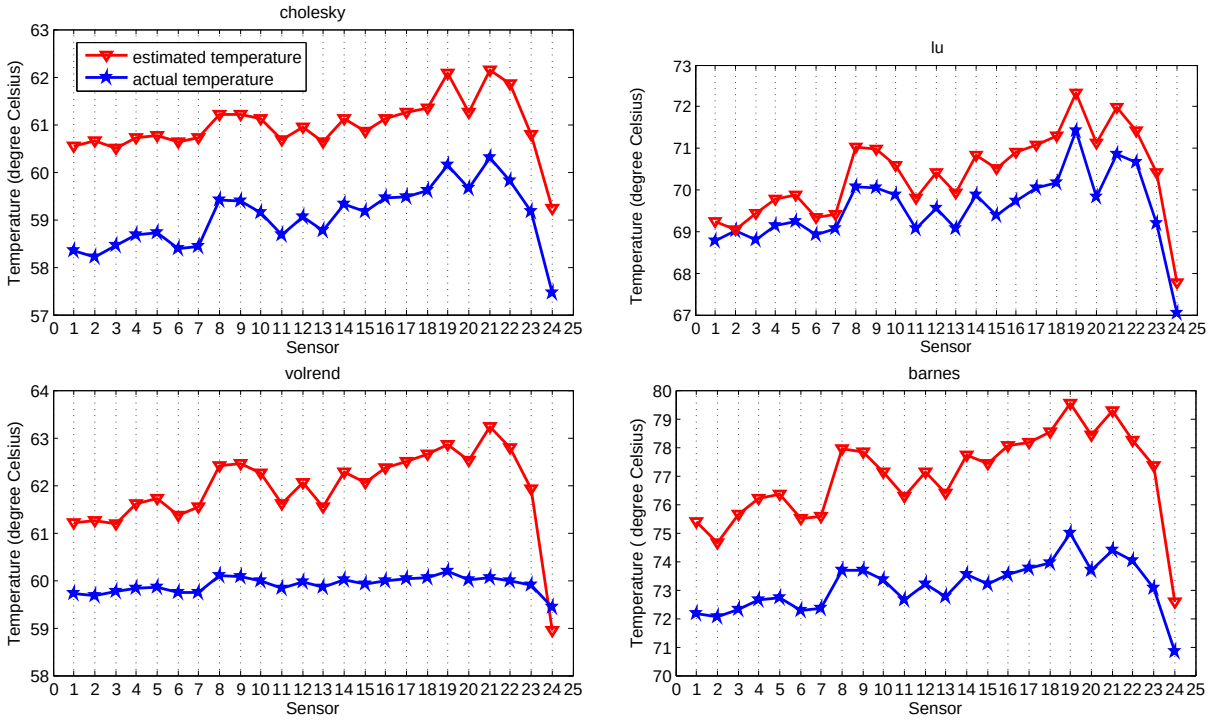

Figure 1: The estimated and actual processor temperature profile at one time instance for four SPLASH2 benchmarks. Each of the 24 thermal sensors in the processor are represented on the horizontal axis for the time instance. More details about the AMD Athlon processor model used for this experimentation are located in Section 5.

obtained in model training, which occurs either during device design or after fabrication once the physical characteristics of the chip have been determined. In the design phase, the performance counter values are obtained from architectural simulators and temperature values are simultaneously generated by performing thermal simulation. The accuracy of the model trained by this method could be limited by the effectiveness of the simulators since they cannot simulate every detail of a real system. In the post-fabrication phase, it is possible to feed workloads to the system and read performance counter registers. At the same time temperature values can be captured through infrared imaging of the running system. Unlike per-chip calibration, it is only necessary to perform data capturing on a small amount of sample chips to get the general information of a particular chip series. We assume that the specific information of an individual chip caused by process variation is reflected in the thermal sensors.

In the training phase, sensor temperatures and performance counter values are recorded at each time instance for a series of time instances. Values $\hat{\beta}_{kj}$ are preliminary model parameters undergoing training. The temperature error of module k at time instance i is

$$e_i^k = \sum_{j=0}^{n} (\hat{\beta}_{kj} M_{ij} - T_i) \tag{7}$$

A least squares regression method minimizes the sum of squares of errors for l time instances:

$$S = \sum_{i=1}^{l} (e_i^k)^2 \tag{8}$$

Equation (9) shows a multi-variable least squares estimator for $\boldsymbol{\beta}$.

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{M}'\mathbf{M}\right)^{-1}\mathbf{M}'\mathbf{T} \tag{9}$$

Here, each column of $\mathbf{M}$ is a time series of a particular performance counter and each column of $\mathbf{T}$ is a time series of temperature at a particular location. $\mathbf{M}'$ is the transposition of $\mathbf{M}$. Once $\hat{\boldsymbol{\beta}}$ is calculated, it can be stored in the programmable registers or re-programmable ROM region of the system.

3.4 Validation of Linear Thermal Estimation Model

To validate the linear model for temperature estimation, SESC and HotSpot are used to obtain performance counter and temperature values, respectively, for an AMD Athlon 64 processor. Temperature values obtained via HotSpot are assumed to be the actual temperatures in this case. The simulator setup is described in more detail in Section 5. Since HotSpot reports the average temperature of a functional unit instead of the temperature at a particular location, we assume that the sensors are located at the center of each functional unit. Although this assumption may not reflect all possible functional unit sensor distributions, the experimental setup still verifies the effectiveness of our

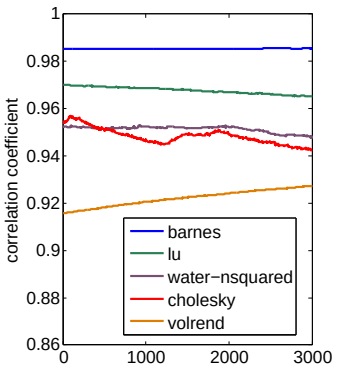

Figure 2: The correlation between estimated and actual temperature profiles

method because the reported temperature reflects local temperature changes and no physical constraints are required for the model.

The SPLASH2 benchmark suite is used to validate the effectiveness of our linear model. The *velosity* benchmark is used to train the linear model (find $\boldsymbol{\beta}$) values and the trained model is tested for temperature estimation on other benchmarks. Figure 1 shows the estimated and actual temperature profile for several benchmarks at a representative time instance. In the subfigures, each point on the x axis represents a thermal sensor value in the processor and there are 24 total sensors integrated on the chip. Sensor 8 and sensor 19 correspond to the integer scheduler and load/store unit, respectively, and they have relatively high temperatures due to high activity. Sensor 24 is in the L2 cache of the processor and its temperature is low because of its large area and relatively low activity.

The estimated temperature profile and the actual temperature profile have very similar shapes in the graphs, so the relative relationship among sensors are estimated correctly. Graphs at other time points are similar. Figure 2 shows the correlation between the estimated and actual temperature profile curves for the benchmarks over a series of 3,000 time points. For all benchmarks, the correlation coefficient is larger than 0.9 which indicates a good linear relationship between the two curves. However, the estimated and actual curves are offset in terms of the absolute temperature value, as shown in Figure 1. Factors such as static power consumption impact the temperature and cannot be estimated with good accuracy by performance counters. In next section, this systematic drift is offset by adding constant values to temperatures determined from sensor readings.

4. MULTI-SENSOR COLLABORATIVE CALIBRATION ALGORITHM (MSCCA)

Resource-limited thermal sensors, such as ring oscillators, often are affected by noise due to process variation and gradual device wear-out. In this section, estimated temperature values obtained in Section 3 and readings taken from sensors are combined using a Bayesian inference based algorithm. The corrected temperatures can then be used for thermal calibration. A comparison of the computational complexity of our method and the KF based method is also presented.

4.1 Problem Formulation

Bayes' theorem presents the relationship between a known (priori) probability distribution and a posterior probability distribution; it is widely used for parameter inference. The unknown parameter distribution is represented by $p(\theta)$, which represents the prior knowledge of θ and the distribution of random variable x for a given θ is $p(x|\theta)$. The distribution of θ after an observation can be calculated using the following formula.

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{10}$$

For our sensor calibration problem, the actual temperatures of sensors are unknown attributes which are estimated by Bayesian inference. The following definitions are used for the formulation of the sensor calibration problem.

$\mathbf{t}$ and $p(\mathbf{t})$: the random vector of the actual temperatures and its probability distribution;
$\mathbf{r}$ and $p(\mathbf{r})$: the random vector of the thermal sensor readings and its probability distribution;
$\mathbf{e}$ and $p(\mathbf{e})$: the random vector of the estimated temperatures and its probability distribution;
$\mathbf{\Sigma_r}$: the covariance matrix of the random vector $\mathbf{r}$;
$\mathbf{\Sigma_e}$: the covariance matrix of the random vector $\mathbf{e}$;
$p(\mathbf{r}|\mathbf{t})$: the probability distribution of the sensor readings given the actual temperatures (sensor noise distribution);
$p(\mathbf{t}|\mathbf{r})$: the probability distribution of the actual temperatures given the sensor readings (statistical inference after an observation);

The probability distribution of the actual temperature $\mathbf{t}$ is given by the following formula. Note that $\mathbf{t}$ and $\mathbf{r}$ are multivariate random variables.

$$p(\mathbf{t}|\mathbf{r}) = \frac{p(\mathbf{r}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{r})} \tag{11}$$

In the above equation, the *priori* knowledge of the actual temperature distribution is $p(\mathbf{t})$, which can be obtained via thermal estimation discussed in the Section 3. So, the *priori* knowledge is $p(\mathbf{e})$. The *posteriori* inference of an actual temperature after an observation is $p(\mathbf{t}|\mathbf{r})$.

Since the temperature change rate is less than 0.1 ^oC per millisecond [15], we assume that the actual temperature keeps constant during a 1 millisecond period. For today's high performance processors, this corresponds to several million clock cycles and enough sensor and performance counter readings can be obtained to perform the calibration algorithm. The corrected temperature is defined as the expected value of the conditional random vector $\mathbf{t}|\mathbf{r}$ which is calculated by the following equation.

$$\boldsymbol{\mu}_t = E(\mathbf{t}|\mathbf{r}) = \int \mathbf{t} \times p(\mathbf{t}|\mathbf{r}) d\mathbf{t} \tag{12}$$

The covariance matrix of the corrected temperature is given as:

$$\mathbf{\Sigma}_t = E[(\mathbf{t} - \boldsymbol{\mu}_t)(\mathbf{t} - \boldsymbol{\mu}_t)'] \tag{13}$$

The probability distribution can be characterized by collecting a time series of sensor readings. Because there are

many factors, such as supply voltage, process variation and ambient temperature fluctuation which impact the sensor readings, the noise of a thermal sensor follows a Gaussian distribution, i.e. $\mathbf{r}|\mathbf{t} \sim \mathcal{N}(\mathbf{t}, \mathbf{\Sigma_r})$. In the Gaussian case, Equations (12) and (13) have a closed form representation as follows [5].

$$\boldsymbol{\mu_t} = \boldsymbol{\mu_e} + \mathbf{\Sigma_e}(\mathbf{\Sigma_e} + \mathbf{\Sigma_r})^{-1}(\mathbf{r} - \boldsymbol{\mu_e}) \quad (14)$$

$$\mathbf{\Sigma_t} = \mathbf{\Sigma_e} - \mathbf{\Sigma_r}(\mathbf{\Sigma_e} + \mathbf{\Sigma_r})^{-1}\mathbf{\Sigma'_e} \quad (15)$$

4.2 Algorithm Description

The Bayesian inference of the actual temperature is used to perform calibration on m thermal sensors once per every p readings (time instances). The steps described in Section 4.1 are performed multiple times per calibration period to refine intermediate results to a final value. In the following description, each algorithm *invocation* is performed on readings from l consecutive time instances. A total of $\frac{p}{l}$ invocations are performed per calibration. The calibration offset for sensor i, w_i, is defined as the difference between the corrected temperature and sensor reading at a specific time point. The $\mathbf{w}$ vector contains all w_i values. The $\mathbf{R}$ matrix ($l \times m$) is initialized with raw sensor data in each invocation and each column represents a time series of readings from one sensor. The $\mathbf{E}$ matrix ($l \times m$) is initialized with raw estimated temperatures in each invocation and each column represents a time series of estimation for one sensor. Step 5 updates the sensors' readings by adding the $\mathbf{w}$ offsets from the previous invocation and Step 6 adjusts the estimated temperatures since these temperatures have systematic error, as mentioned in Section 3.4. The value c is the mean value of all elements in $\mathbf{R}$. Overall, algorithm 1 shows the multi-sensor collaborative calibration algorithm using Bayesian inference over multiple invocations until all p readings for m sensors have been processed.

Algorithm 1 Multi-Sensor Collaborative Calibration Algorithm – MSCCA

1: Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$.
2: **while** Invocation count $\leq \frac{p}{l}$ **do**
3: Store sensor readings in $\mathbf{R}$ matrix for next l time instances.
4: Store estimated temperatures determined from approach in Section 3 in $\mathbf{E}$ matrix.
5: Adjust $\mathbf{R}$ matrix by adding offset $\boldsymbol{w}$ to each row.
6: Adjust $\mathbf{E}$ matrix by subtracting a constant value c.
7: The vector $\mathbf{r}$ is the columnwise mean of $\mathbf{R}$.
8: The vector $\boldsymbol{\mu}_e$ is the columnwise mean of $\mathbf{E}$.
9: Calculate the covariance matrices $\mathbf{\Sigma_r}$ and $\mathbf{\Sigma_e}$.
10: Perform Bayesian inference using Equations (14) and (15), and get the corrected temperature $\boldsymbol{\mu_t}$.
11: $\boldsymbol{w} \leftarrow \boldsymbol{\mu_t}$ - $\mathbf{r}$.
12: **end while**

4.3 Computational Complexity Evaluation

This section analyzes the computational complexity of the MSCCA and compares it with the complexity of KF based approaches. Table 3 shows the number of operations performed by the two approaches for p sets of readings. For the MSCCA approach, l time instances (sets) of readings per invocation are used. As noted in the previous subsection, calibration can be simultaneously performed for multiple consecutive sensor readings for each sensor in one MSCCA invocation. In contrast, KF based algorithms predict and update the temperature for each set of sample readings, resulting in more matrix operations. In our implementation, there are $m = 24$ thermal sensors, so the matrix dimensions of Σ_r and Σ_e are 24×24. If the matrix operations are converted to scalar operations, there are about $140,000p$ additions and $140,000p$ multiplications required for the KF method. In our method, the numbers of additions and multiplications are about $\frac{p}{l}(444l + 14,000)$ and $\frac{p}{l}(300l + 14,000)$. Both algorithms run in $O(p)$ time complexity. Since samples must be stored in matrices for a period of time before they are processed, MSCCA does require more memory usage than the KF based approach.

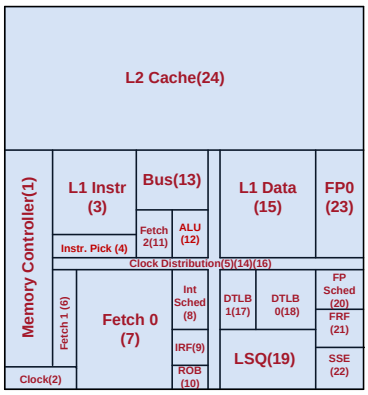

Figure 3: Floorplan of the Athlon 64 processor [10]

Table 3: Operations required by MSCCA and Kalman filtering for p sets of sample readings for 24 sensors

Operation	MSCCA	KF scheme
scalar addition	$\frac{p}{l}(444l - 48)$	0
scalar multiplication	$\frac{p}{l}(300l + 48)$	0
matrix addition	$\frac{p}{l}$	$2p$
matrix multiplication	$\frac{p}{l}$	$10p$
matrix-vector multiplication	$\frac{p}{l}$	$3p$
matrix inversion	$\frac{p}{l}$	p
vector addition	$\frac{p}{l}$	$3p$

5. EXPERIMENTAL APPROACH

The proposed calibration strategy is verified via simulation for an AMD Athlon 64 processor. The floorplan of AMD Athlon 64 processor is shown in Figure 3. The processor includes 24 functional blocks, which are labeled in the figure. After technology scaling, the area of this processor is estimated to be $3.6mm^2$ in 45nm technology. The frequency of the processor is configured at 1GHz and the overall initial temperature of the processor is set to 50^oC.

The SESC simulator integrated with HotSpot is used to generate performance counter and temperature values. The SESC simulator simultaneously generates system statistics and power traces every 0.1 millisecond for each benchmark, and the floorplan of the processor and generated power trace are fed into HotSpot to generate temperature traces. We assume that the HotSpot generated temperature values are the

actual temperatures considering the sophisticated thermal model implemented by HotSpot, and random noise is added to these actual temperatures to get the simulated thermal sensors readings. The standard deviation of random temperature noises is set from $1^oC \sim 8^oC$. We assume no specific type of on-chip thermal sensor implementation.

The MSCCA algorithm described in Section 4 was implemented in MATLAB. From our generated data, the temperature change rate is less than $0.01^oC/ms$. We took 100 $\sim$ 1000 consecutive time instances as the algorithm inputs so that the actual temperature change in one invocation is limited in 1^oC. For comparison, we also implemented the temperature characterization algorithm based on KF [18]. The power profile statistics which are used in KF were captured using the power trace generated by SESC. A linear regression method was used to obtain thermal resistance and capacitance parameters based on the differential equation for heat diffusion.

6. RESULTS

Results were obtained using applications from the SPLASH2 benchmark suite executed with the SESC simulator. The thermal estimation model described in Section 3 was trained using the *radiosity* workload and the remainder of the benchmarks were used to verify our MSCCA algorithm. Although SPLASH2 supports multi-threaded simulation for multi-core systems, all benchmarks are configured to run in a single processor in our experiments.

6.1 Effectiveness Verification

In a first series of experiments, the thermal profiles of four SPLASH2 benchmarks for the AMD Athlon 64 are determined. For these experiments, the standard deviation of noise is set to 4 ^oC and p=2000 total time instances of readings are processed. MSCCA uses l=100 time instances per invocation. In Figure 4, we demonstrate the thermal profile of the AMD Athlon 64 processor for the *lu* benchmark after the 2000th time instance. The horizontal axis represents thermal sensors for each functional block in Figure 3. In the figure, the actual temperature, sensor readings, and corrected temperature from the KF based implementation and from MSCCA are plotted. Both constant spatial noise due to process variations and temporal noise (shown in the plot of the sensor readings) are taken into account in our simulation. The first observation is that both methods effectively reduce the sensor reading errors: the sum of the square errors of all sensors for the corrected readings is much smaller than that of sensor readings. The second observation is that the thermal profile is recovered after synthesizing two data sources (the estimated temperature and sensor readings in our case, the statistical characteristics of the power dissipation and sensor readings in the KF case).

Figure 5 shows the standard deviations of the temperature errors for six benchmarks over 2000 time instances (100 time instances per invocation for MSCCA). There are four temperature calculation types plotted in the figure. The error deviations are significantly reduced after performing our algorithm and KF for all benchmarks. Our method and the KF based method have comparable accuracy results. In some cases the error deviation of our algorithm is smaller than that of KF based approach. In other cases, the KF

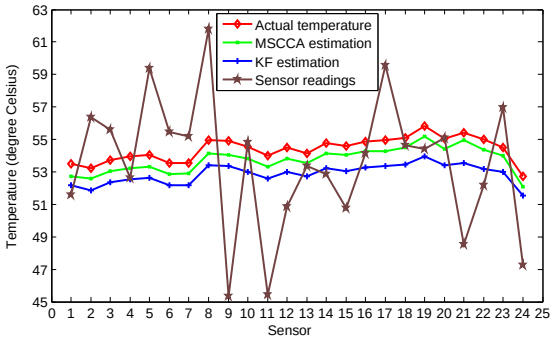

Figure 4: The thermal profile of the processor for one time instance of the *lu* benchmark. Each point on the horizontal axis represents a single sensor located in a block in Figure 3.

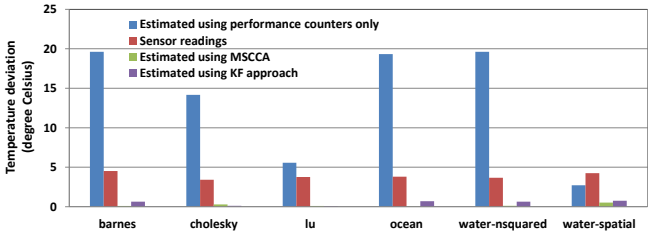

Figure 5: Standard deviation of the temperature error for SPLASH2 benchmarks

based method outperforms our method. Overall, the error deviations are reduced to $\leq 1^oC$ for all benchmarks over the 2000 time instances.

6.2 Run Time Comparison

The run time of the MSCCA and KF approaches were measured in MATLAB. The run time results are presented in Table 4. In this table, we consider the total running time for both approaches over p=10,000 readings (time instances) for each sensor. As mentioned in Section 4.3, the inner loop of MSCCA can be invoked once per l time instances, while KF is invoked for each time instance. The first row of the table indicates the total number of time instances for each invocation in the MSCCA algorithm, and there are 10,000 time instances in total for each case. For example, if it processes l=200 time samples of values stored in $\mathbf{R}$ and $\mathbf{E}$ matrices per invocation, the algorithm is invoked 50 times for 10,000 time instances. From the table, MSCCA takes 0.0283 seconds to finish 50 invocations (10,000 total time instances per sensor) while the KF approach requires approximately 67x more time. The run time ratio increases as the number of time samples processed per MSCCA invocation increases.

6.3 Impact of Sensor Reading Errors

The standard deviation of the errors of the corrected temperature increases as the noise of sensor readings become larger. The experiments in the previous subsection were repeated, this time with varying amounts of noise in the sensor readings. Experiments of 10,000 time instances each were performed. Table 5 shows the standard deviations of

Table 4: Run time comparison (in seconds) between MSCCA and KF approaches for 10000 time instances

Time instances per invocation	200	400	1000
MSCCA run time	0.0283	0.0223	0.0161
KF run time	1.9146	1.9049	1.8033
Run time ratio	67	85	112

Table 5: The standard deviation of the error for the corrected temperatures over 10,000 time instances for increasing sensor error

Time instances	Std dev of sensor error ^oC			
per invocation	2	4	6	8
200	0.327	0.358	0.412	0.453
400	0.318	0.380	0.408	0.384
600	0.271	0.360	0.353	0.406
800	0.346	0.362	0.393	0.422
1000	0.342	0.367	0.366	0.411

corrected temperatures for sensor readings with four different sensor noise levels. As predicted, the less accurate the sensor readings are, the larger error seen in the corrected temperature.

7. CONCLUSION

In this paper, two contributions regarding the real-time calibration of thermal sensors are provided. First, we describe an approach to directly use information from performance counters to estimate temperature for specific on-chip functional units without using intermediate power estimation. The technique uses a collection of parameters that can be determined at design time or after chip fabrication via model training with a collection of application benchmarks. Second, we present an algorithm (MSCCA) based on Bayesian inference to combine these estimated temperatures and temperature readings sampled from sensors into corrected readings. The effectiveness of the proposed algorithm is validated and the results show that the error and its standard deviation are reduced to $\leq 1^oC$. This value is consistent with current state-of-the-art Kalman filtering based approaches which take much longer to calculate.

8. REFERENCES

[1] Hotspot. http://lava.cs.virginia.edu/HotSpot/.

[2] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *International Symposium on High Performance Computer Architecture*, pages 171 –182, Jan. 2001.

[3] R. Cochran and S. Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *ACM/IEEE Design Automation Conference*, pages 478–483, July 2009.

[4] B. Datta and W. Burleson. Calibration of on-chip thermal sensors using process monitoring circuits. In *International Symposium on Quality Electronic Design*, pages 461 –467, March 2010.

This work was funded by the Semiconductor Research Corporation under Task 2083.001

[5] E. Elnahrawy and B. Nath. Cleaning and querying noisy sensors. In *International Conference on Wireless Sensor Networks and Applications*, pages 78–87, Sept. 2003.

[6] H. F. Hamann, A. Weger, J. A. Lacey, Z. Hu, P. Bose, E. Cohen, and J. Wakil. Hotspot-limited microprocessors: Direct temperature and power distribution measurements. *IEEE Journal of Solid-State Circuits*, 42(1):56–65, Jan. 2007.

[7] A. Kumar, L. Shang, L.-S. Peh, and N. Jha. System-level dynamic thermal management for high-performance microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):96 –108, Jan. 2008.

[8] K.-J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *IEEE International Parallel and Distributed Processing Symposium*, page 232, April 2005.

[9] F. Liu. A general framework for spatial correlation modeling in VLSI design. In *IEEE/ACM Design Automation Conference*, pages 817–822, June 2007.

[10] F. J. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau. Power model validation through thermal measurements. In *International Symposium on Computer Architecture*, pages 302–311, June 2007.

[11] M. Powell, A. Biswas, J. Emer, S. Mukherjee, B. Sheikh, and S. Yardi. CAMP: A technique to estimate per-structure power at run-time using a few simple parameters. In *IEEE International Symposium on High Performance Computer Architecture*, pages 289–300, Feb. 2009.

[12] S. Reda. Thermal and power characterization of real computing devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(2):76 –87, June 2011.

[13] S. Remarsu and S. Kundu. On process variation tolerant low cost thermal sensor design in 32nm CMOS technology. In *ACM Great Lakes Symposium on VLSI*, pages 487–492, May 2009.

[14] J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[15] S. Sharifi and T. Rosing. Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(10):1586–1599, Oct. 2010.

[16] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *ACM International Workshop on Wireless Sensor Networks and Applications*, pages 59–67, Sept. 2002.

[17] W. Wu, L. Jin, J. Yang, P. Liu, and S.-D. Tan. A systematic method for functional unit power estimation in microprocessors. In *ACM/IEEE Design Automation Conference*, pages 554 –557, July 2006.

[18] Y. Zhang and A. Srivastava. Adaptive and autonomous thermal tracking for high performance computing systems. In *ACM/IEEE Design Automation Conference*, pages 68 –73, June 2010.