Dynamic On-Chip Thermal Sensor Calibration Using Performance Counters

Shiting (Justin) Lu, Russell Tessier, *Senior Member, IEEE,* and Wayne Burleson, *Fellow, IEEE*

***Abstract*—Numerous sensors are currently deployed in modern processors to collect thermal information for fine-grained dynamic thermal management (DTM). To enhance processor performance and avoid overheating, accurate temperatures must be obtained from thermal sensors. Due to process variation and silicon aging, on-chip thermal sensors require periodic calibration before use in DTM. However, the calibration cost for thermal sensors can be prohibitively high as the number of on-chip sensors increases. In this work, a model which is suitable for on-line calculation is employed to estimate the temperatures of multiple sensor locations on the silicon die. The estimated sensor and actual sensor thermal profile show a very high similarity with correlation coefficient ~ 0.9 for most tested benchmarks. Our calibration approach 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 average absolute error of the corrected sensor temperature readings is $< 1.5^oC$ and the standard deviation of error is less than $< 0.5^oC$ for tested benchmarks.**

***Index Terms*—thermal sensors, performance counters, DTM, thermal estimation, sensor calibration**

I. Introduction

THERMAL management is a critical problem for modern microprocessors 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 [1]. On-chip thermal sensors deployed in microprocessors are currently used to assist DTM. For example, there are five thermal sensors per core implemented in the Power 7 EnergyScale infrastructure [2] and 12 sensors on each CPU core in the Intel Sandy Bridge [3]. Recent trends indicate increased future use to assess thermal gradients and perform fine-grained thermal management with more on-chip thermal sensors.

The efficiency and effectiveness of a DTM strategy relies on accurate thermal sensor measurements. However, on-chip thermal sensors are sensitive to process variations and can report temperature values which deviate from actual ones. A popular implementation of an on-chip thermal sensor uses a ring oscillator whose frequency can be mapped to a temperature value. In some cases, the temperature reading error of *uncalibrated* thermal sensors can be substantial (up to 34^oC at 95^oC [4]) which adversely impacts DTM strategy. 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 [5]. 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. Even if thermal sensors are initially well-calibrated, their readings gradually drift away from actual temperature values due to device wear-out. Often, the degree of aging varies across the chip due to the activity variation of different subcircuits. Therefore, recalibration on thermal sensors 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.

In this paper, we take a different approach to temperature estimation which is optimized for on-line calibration of thermal sensors. To dynamically account for changing activities of the processor, collections of performance counter values are used to estimate the chip thermal profile at run time. A performance counter selection method is employed to reduce the intercorrelations between readings and improve estimation accuracy. Our results show that the correlation coefficient between estimated and actual thermal profiles is above 0.95 on a collection of benchmarks. In this paper, *estimated temperature* is exclusively used to refer to a temperature obtained from thermal estimation using performance counters.

Estimated temperature values derived from performance counter values are used to validate and correct divergent sensor readings via a multi-sensor collaborative calibration algorithm (MSCCA). This algorithm can be executed at run time using a block of consecutive sensor readings. *Corrected temperature* values obtained from the algorithm are then used to adjust the mapping of thermal sensor parameters to temperature readings. A Bayesian technique integrated into MSCCA utilizes the implicit physical proximity of the estimated temperature locations (spatial correlation) to correct sensor reading errors. An increased number of on-chip sensors provides increased algorithm accuracy since more spatial correlation information is captured within the estimated temperatures.

To validate our algorithm, architectural and thermal simula-

This work was supported by the Semiconductor Research Corporation under Task 2083.001. S. Lu, R. Tessier and W. Burleson are with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, 01003, USA e-mail: jlu, tessier, burleson@ecs.umass.edu.

tors are used to collect performance counter and temperature values, respectively. Results show that the average absolute error of thermal sensor readings can be effectively reduced to $\leq 1.5^oC$ for all tested benchmarks and the standard deviation of the errors in the corrected temperatures is reduced to an acceptable level (from $3 \sim 4^oC$ to $\leq 0.5^oC$). For comparison, the computational complexity and estimation accuracy of our new approach is evaluated against a method which is based on power estimation and Kalman filtering [6]. Our results show that the new strategy has comparable accuracy (better in many cases) but is more efficient (at least $50\times$ faster).

II. Background

There are several ways to calibrate on-chip thermal sensors to achieve better measurement accuracy. We divide different approaches into three main categories: thermal imaging, design for calibration, and thermal estimation.

A. Thermal Imaging

The 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 [7]. The model parameters can then be obtained by solving a series of equations or by using statistical parameter inference [8]. 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. Although the approach could be used once after chip fabrication, it cannot be used effectively at run time.

B. Design for Calibration

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 [9]. 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.

C. Thermal Estimation

A third approach uses accurate on-chip thermal estimation instead of thermal imaging to determine actual temperatures. This information can then be used for calibration of specific sensors. In Liu [10] and Cochran and Reda [11], the authors describe methods to construct thermal estimates for numerous points in a processor from measurement data from a sparse set of thermal sensors. In Ranieri *et al.* [12], the overall thermal map is recovered from a reduced set of sensors by selecting principal eigenvectors of the whole-chip temperature vector. In Zhou *et al.* [13], an information-theoretic framework is proposed to find the optimal location for sensor deployment and full-chip thermal monitoring. Since the thermal sensor measurements are subject to noise, the amount of error at each specific 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 generate estimates of actual temperature.

Kumar *et al.* [14] 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 , so an estimation strategy with finer granularity is needed. Lee *et al.* [15] 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 [6][16], two sources of temperature information are combined to generate temperature estimates: (1) noisy sensor readings and (2) localized power consumption which is 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 [17], 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 [18]. 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. In Zhang and Srivistava [19], the temperature for noisy sensors is estimated using statistics.

Like other approaches in the previous paragraph, our new calibration method 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 estimated temperature. This direct approach reduces run time and eliminates the need to estimate per-functional unit power consumption. A merging algorithm is used to combine our temperature estimates and sensor reading data.

This work greatly expands upon the material in our previous ICCAD'2012 paper [20] which targeted similar goals. In this paper we track incremental Δ changes in temperature rather than absolute temperature, dramatically increasing accuracy. Also, we consider temperature gradients between sensors based on earlier estimated temperatures in addition to performance counter values in determining current temperature estimates for sensors. To validate the effectiveness of the approach, 15 benchmarks are used to perform temperature verification and the training process for our model development is performed using 16 benchmarks, rather than one.

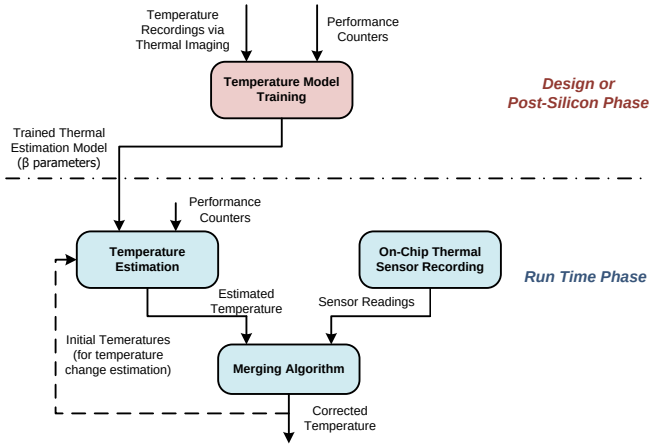

Fig. 1. Our dynamic calibration scheme for on-chip thermal sensors

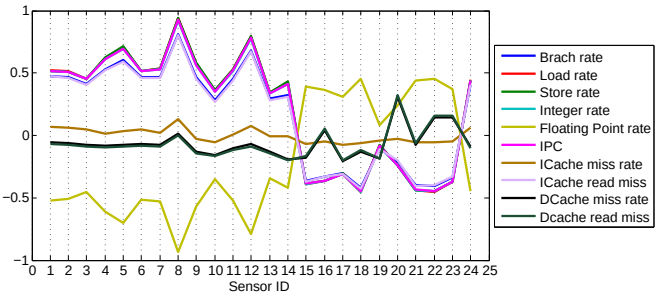

Fig. 2. Correlation between temperature and some system statistics for the *radix* benchmark across 24 thermal sensors

D. Our Calibration Strategy: Approach Overview

Fig. 1 shows our strategy for dynamic on-chip sensor calibration. This flow can be broken down into four steps, one which is performed once during the design or post-silicon phase and three which are performed repetitively at run time.

1) **Temperature model training** - In the design or post-silicon phase, a thermal estimation model is developed based on accurate temperature recordings through thermal imaging technology and system statistics from performance counters. The model training outputs a set of parameters called β parameters. These β parameters define the relationship between performance counter values and estimated temperatures.
2) **Temperature estimation** - The β parameters are used in a series of linear equations to convert performance counter values to temperature estimates. Although useful, temperatures obtained from this model often do not meet accuracy requirements since performance counters cannot capture all on-chip thermal details precisely.
3) **On-chip thermal sensor recording** - Potentially noisy thermal sensor readings are collected from on-chip thermal sensors.
4) **Merging algorithm** - To calibrate a thermal sensor, we combine thermal estimations and sensor readings using a Bayesian-based fusion algorithm. This *MSCCA algorithm* generates *corrected* temperature values and identifies how much a thermal sensor should be adjusted in calibration, if needed.

In the following section, temperature estimation and model training are considered. Section IV describes our merging algorithm and the techniques used for on-chip sensor reading.

III. Temperature Estimation and Model Training Using Performance Counters

In a microprocessor, performance counters monitor runtime 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 granularity using linear regression [17] or unit power consumption [15][21]. Unlike power consumption estimation, functional unit temperature estimation is more complex due to inter-component correlation resulting from heat flow across the chip.

It can be shown that the temperatures of functional units are correlated with values read from on-chip performance counters. Fig. 2 shows the correlation coefficients of component temperatures and various system statistics for the SPLASH-2 *radix* benchmark [22]. Most temperature-statistic pairs show non-zero correlation coefficients. For example, the floating point rate shows a negative correlation with integer units (e.g. the integer scheduler) and a positive correlation with floating point components (e.g. the floating point scheduler).

To explore the correlation between the application characteristics and temperature changes over a short time period, system statistics and a temperature trace were recorded for every millisecond via simulation using SESC [23] and HotSpot [24]. Fig. 3 illustrates the relation between the system statistics: IPC (the first row), integer instruction rate (the second row) and floating point instruction rate (the third row), and temperature change rates for three functional units: integer scheduler (left column), floating point scheduler (middle column) and L1 data cache (right column). The performance counter data was obtained by repeatedly running the *equake* benchmark from the SPEC2000 benchmark suite using the SESC simulator. The temperature trace was generated by HotSpot.

As seen in the figure, higher average IPC (phase A in the top-left sub-figure) results in a higher temperature change rate for the integer scheduler at the start of phase A. However, this change rate is negative for the floating point scheduler at the same point. The floating point instruction rate is higher in phase B and the scheduler is more active during this phase. leading to a floating point scheduler instruction temperature surge every time the application transitions from phase A to phase B, although it is short. In this case, the temperature of the component is impacted by its surroundings due to heat flow.

The temperature difference (referred to as thermal gradient in the figure) between the specific component and a neighboring component is plotted to illustrate this point in Fig. 4. The thermal gradient in the figure is obtained by taking the maximum temperature difference among all neighboring blocks. In the integer scheduler, for example, the temperature surge

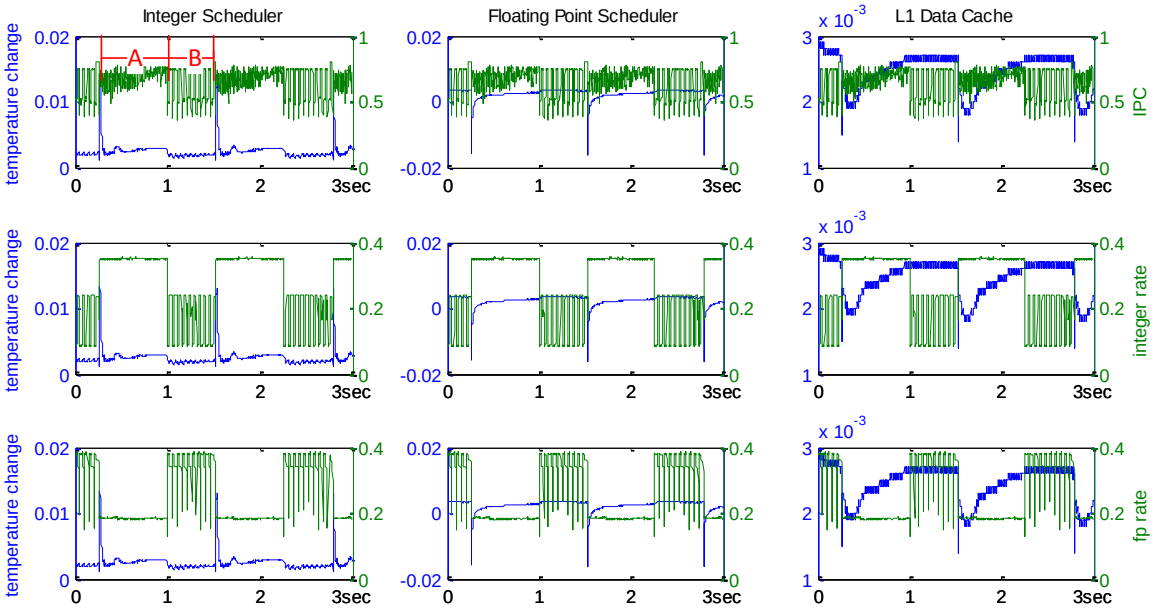

Fig. 3. Runtime recording of some system statistics and temperature change rates for three function units: integer scheduler (left column), floating point scheduler (middle column) and L1 data cache (right column) for the *equake* benchmark. Temperature changes are in ^oC

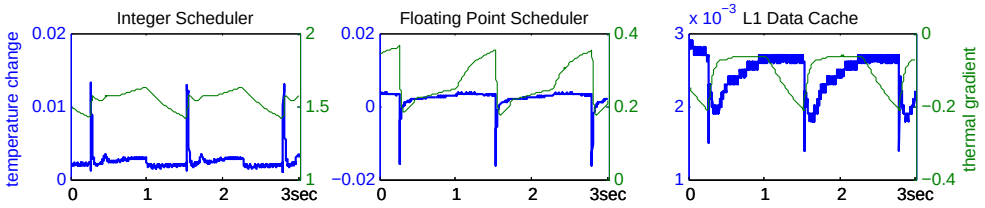

Fig. 4. Runtime recording of thermal gradient and temperature change rates for three function units. Temperature changes are in ^oC

causes an increase in the thermal gradient which accelerates heat flow from the integer scheduler to its neighbors. A new thermal balance is quickly reached after a short time.

A. Temperature Estimation Using Performance Counters

By virtue of a variety of inter-unit thermal complexities, the temperature at a specific position on the chip is generally not linearly related to one particular performance counter. However, we demonstrate that a linear model can be used to estimate on-chip temperature changes at specific temperature sensors using multiple performance counters if the time interval is small. This linear approximation is shown to be effective empirically in developing an on-chip thermal profile. These thermal estimates can then be merged (Section IV) with sensor readings to reduce spatial (across sensors) and temporal (across time) noise. In the following derivation we are interested in determining ΔT estimates for specific sensors over a time interval, rather than absolute T values.

1) Linear Temperature Estimator: In developing a linear model for a specific thermal sensor i over a time interval, a row vector ($\mathbf{x^i}$) contains recorded performance counter values for the interval, $\mathbf{M}$ (listed in Table I), thermal gradient information, $\mathbf{G^i}$, and temperature, T^i. Values $\mathbf{G^i}$ measure the thermal gradient between other thermal sensors and sensor i at the beginning of the interval. Value T^i measures the temperature of sensor i at the beginning of the interval. Since the correct temperature before the first interval is unknown, the thermal gradients and T^i can be approximated by using thermal sensor readings at the start of calibration. For other intervals, the temperature estimation from the previous interval is used. The combination of $\mathbf{M}$, $\mathbf{G^i}$, and T^i forms $\mathbf{x^i}$:

$$\mathbf{x^i} = [\mathbf{M}, \mathbf{G^i}, T^i] \quad (1)$$

For example, for the integer scheduler (unit 8), $\mathbf{G^8}$ includes all temperature differences between the integer scheduler and other components at the beginning of the measurement interval. Here, the superscripts of T indicate the hardware components in the floorplan (Fig. 10 in Section V). The thermal gradient vector for the integer unit is given by (2). There are 24 architectural components in the studied processor,

TABLE I
PERFORMANCE COUNTERS PROVIDED BY SESC

general rate	IPC, integer rate, load rate store rate, floating point rate
cache	Dcache read miss rate, Dcache write miss rate, Icache miss rate
buffer and queue usage	load queue, store queue, ROB, Iwin, TLB
branch	BTB, RAS

so $\mathbf{G^8}$ contains 23 elements which are temperature differences between the components and the integer scheduler, except itself. The performance counter vector $\mathbf{M}$ can be represented by (3), where u is the number of performance counters used in the model.

$$\mathbf{G^8} = [T^1\text{–}T^8, ..., T^7\text{–}T^8, T^9\text{–}T^8, ..., T^{24}\text{–}T^8] \quad (2)$$

$$\mathbf{M} = [P^1, P^2, ..., P^u] \quad (3)$$

The T^i value in (1) is used to take static power (which is dependent on temperature) into account. Therefore, the sampled vector at a particular time step is given by (4) for the integer scheduler. It is apparent that both hardware activities (as measured by performance counters) and thermal gradients impact temperature change during the sampling interval.

$$\mathbf{x^8} = [P^1, ..., P^u, T^1\text{–}T^8, ..., T^7\text{–}T^8, T^9\text{–}T^8, ..., T^{24}\text{–}T^8, T^8] \quad (4)$$

Performance counter values represent *changes* in the respective event counters during the sampling interval. Only events happening in a specific interval are evaluated for the corresponding performance counter monitors. Using the above $\mathbf{x}$ vector, it is possible to estimate the temperature change of a particular component which contains the thermal sensor during the sampling interval using a linear equation. For example, the equation for thermal sensor i is:

$$\Delta T^i = \mathbf{x^i} \cdot \boldsymbol{\beta}^i \quad (5)$$

and for the sensor in the integer scheduler:

$$\Delta T^8 = \mathbf{x^8} \cdot \boldsymbol{\beta}^8 \quad (6)$$

Here, $\boldsymbol{\beta}^8$ is a column vector whose elements are coefficients of the linear model. The coefficient vector $\boldsymbol{\beta}^8$ can be determined through model training which will be discussed in the next subsection. Each sensor i is trained separately to obtain its own $\boldsymbol{\beta}$ coefficient vector. The linear model only needs multiplications and additions to calculate the results, so the time cost is low and calculations can be done in real time.

B. Linear Model Training

As mentioned earlier in this section, the first step in developing a relationship between performance counter values and estimated temperatures (e.g. $\boldsymbol{\beta}$ vectors) involves training. The accuracy of the coefficient vector $\boldsymbol{\beta^i}$ impacts the model accuracy for sensor i. In this training step, accurate known temperatures for the sensors must be available to develop the relationships. As mentioned in Section II, these relationships can be determined via architectural and thermal simulation during design once physical characteristics of the chip have been determined or during post-fabrication testing using thermal imaging. In post-fabrication testing, it is possible to feed real 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 following training analysis, we assume that accurate temperatures and $\mathbf{x}$ values consisting of $\mathbf{M}$ and $\mathbf{G}$ are available for all sensors. The most straightforward way to train the linear model to determine $\boldsymbol{\beta}$ vectors in (5) is to use an ordinary least square method (OLS) [20]. The coefficient vector obtained through OLS is given by (7).

$$\boldsymbol{\beta^i}_{ols} = (\mathbf{X^i}^\mathbf{T}\mathbf{X^i})^{-1}\mathbf{X^i}^\mathbf{T}\mathbf{y^i} \quad (7)$$

Here, $\mathbf{X^i}$ is a matrix consisting of row vectors $\mathbf{x^i}$ calculated over a series of $\mathbf{N}$ sampling intervals. Each row of $\mathbf{X^i}$ represents $\mathbf{x^i}$ for one sample interval. $\mathbf{y^i}$ is a column vector comprised of accurate *actual* temperature changes for sensor i which occur during the respective training intervals. Although OLS is capable of training the linear model, its somewhat simplistic formulation does not consider the intercorrelation of performance counters, limiting accuracy.

A more advanced, iterative mathematical approach can be used to determine $\boldsymbol{\beta}$ values. As an alternative to OLS, we use automatic relevance determination (ARD), which was developed by MacKay [25] and Neal [26]. The coefficient vector $\boldsymbol{\beta}^i$ for sensor i can be represented by the following expressions (8) and (9).

$$\boldsymbol{\beta}^i = \delta^{-2}\mathbf{S}\mathbf{X^i}^T y^i \quad (8)$$

$$\mathbf{S} = \left(\mathbf{A} + \delta^{-\mathbf{2}}\mathbf{X^i}^\mathbf{T}\mathbf{X^i}\right)^{-1} \quad (9)$$

In (9), $\mathbf{A} = \text{diag}(\alpha_1, ..., \alpha_M)$, which is a diagonal matrix. Each α_j in $\mathbf{A}$ represents the relevance of an input vector variable to the result such that:

$$\alpha_j = \frac{1 - \alpha_j {S_{jj}}^2}{\beta_j^i} \quad (10)$$

$$\delta^2 = \frac{\sum_{n=1}^{N}(y_n^i - {x^i}_n \cdot \beta_n^i)^2}{N - \sum_{j=1}^{M}(1 - \alpha_j S_{jj})} \quad (11)$$

Since (8) and (9) depend on (10) and (11) and vice versa, multiple iterations are needed to achieve convergence of the β^i unknowns. These iterations calculate α_j in diagonal matrix $\mathbf{A}$ and δ. In the above equations, S_{jj} are elements of $\mathbf{S}$. y_n is the n^{th} element of $\mathbf{y^i}$ and $\mathbf{x}_n$ is the n^{th} row vector of $\mathbf{X^i}$. N is the number of training samples and M is the length of vector $\mathbf{x^i}$ and the dimension of the matrix $\mathbf{S}$. Values for α_j and δ^2 are determined by alternating evaluation of the above four equations until convergence. From our experiments, around four iterations are performed until these parameters reach convergence.

C. Evaluation of Model Training Accuracy Using Thermal Estimation

To evaluate the accuracy of the linear model, Tables II and III show the estimation error for one time interval. In this case, (5) is evaluated for one time interval, using known T values to determine $\mathbf{G}$ gradients and measured performance counter values $\mathbf{P}$. Errors between the actual ΔT and ΔT values determined with (5) are then calculated. Table II gives

TABLE II
AVERAGE ESTIMATION ERROR FOR EACH SENSOR OVER ALL BENCHMARKS

Sensor ID	1	2	3	4	5	6	7	8	9	10	11	12
Avg. abs. error (^oC)	0.0003	0.0002	0.0003	0.0002	0.0002	0.0002	0.0002	0.0035	0.0003	0.0002	0.0002	0.0014
Std. abs. error (^oC)	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0045	0.0002	0.0002	0.0002	0.0013
Avg. change (^oC)	0.0011	0.0010	0.0078	0.0081	0.0069	0.0010	0.0022	0.0838	0.0288	0.0222	0.0046	0.0274
Std. change (^oC)	0.0007	0.0005	0.0037	0.0043	0.0047	0.0007	0.0016	0.0272	0.0122	0.0096	0.0035	0.0090
Sensor ID	13	14	15	16	17	18	19	20	21	22	23	24
Avg.abs. error (^oC)	0.0002	0.0003	0.0002	0.0004	0.0003	0.0003	0.0016	0.0072	0.0003	0.0015	0.0014	0.0005
Std. abs.error (^oC)	0.0002	0.0002	0.0002	0.0003	0.0002	0.0002	0.0008	0.0049	0.0003	0.0010	0.0009	0.0003
Avg. change (^oC)	0.0166	0.0133	0.0032	0.0094	0.0139	0.0140	0.0102	0.0437	0.0139	0.0126	0.0120	0.0001
Std. change (^oC)	0.0088	0.0088	0.0024	0.0058	0.0078	0.0076	0.0056	0.0119	0.0060	0.0049	0.0044	0.0001

TABLE III
AVERAGE ESTIMATION ERROR FOR EACH BENCHMARK OVER ALL SENSORS

benchmark	mcf	vortex	swim	art	apsi	radiosity	ocean	radix
Avg. abs. error (^oC)	0.0017	0.0005	0.0008	0.0019	0.0006	0.0006	0.0011	0.0032
Std. abs. error (^oC)	0.0022	0.0003	0.0009	0.0004	0.0006	0.0007	0.0009	0.0010
Avg. change (^oC)	0.0082	0.0117	0.0365	0.0338	0.0048	0.0086	0.0409	0.0078
Std. change (^oC)	0.0002	0.0116	0.0058	0.0002	0.0004	0.0045	0.0123	0.0060
benchmark	parser	twolf	vpr	ammp	applu	barnes	fft	water-spatial
Avg.abs. error (^oC)	0.0004	0.0002	0.0003	0.0005	0.0005	0.0007	0.0008	0.0004
Std. abs. error (^oC)	0.0007	0.0001	0.0004	0.0006	0.0010	0.0003	0.0010	0.0004
Avg. change (^oC)	0.0113	0.0036	0.0068	0.0047	0.0178	0.0046	0.0195	0.0181
Std. change (^oC)	0.0036	0.0031	0.0046	0.0006	0.0178	0.0022	0.0112	0.0172

the average absolute error and error standard deviation for each sensor for a single interval using the β values determined through training. The error is averaged over all 16 test benchmarks (benchmarks described in more detail in Section V). Using information from this table it is possible to evaluate the trained models for all sensors. Sensor 8 (integer scheduler) and sensor 20 (FP scheduler) report relatively high error and error variation compared with other sensors due to their high activity. The error can potentially be reduced further with a smaller time step, but this approach increases the computational cost correspondingly. Table III gives the average absolute error and the associated standard deviation for each benchmark. The error is averaged over all sensors for each benchmark. From this table, we can evaluate how the trained models work for all benchmarks. In general, average absolute error and standard deviation are low in the tables. During experimentation we found that the trained model was most effective for benchmarks which have similar execution characteristics to the benchmark training set. However, the use of a broad class of benchmarks for training helps minimize error across a larger number of benchmarks.

D. Thermal Estimation Results

The SPLASH-2 and SPEC2000 benchmark suites were used to validate the effectiveness of our linear model. Mixed samples from a subset of benchmarks were used to train the linear model (find β values) and the trained model was tested for estimation on the rest of the benchmarks. Detailed simulator setup information is provided in Section V.

1) Estimated Thermal Profile: Fig. 5 shows the thermal profile across all sensors at four time instances of the

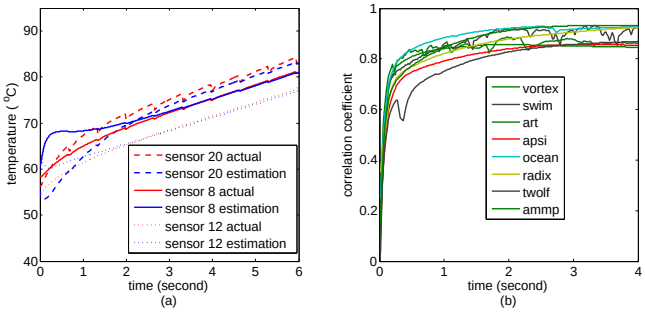

Fig. 6. (a) Temperature estimation over 6 seconds for sensor 8 (integer scheduler), 20 (floating point scheduler) and 12 (ALU). The data is collected through simulation by running *applu* on the AMD floorplan. (b) The correlation between estimated and actual temperature profiles for various benchmarks.

SPLASH-2 *ocean* benchmark. Experiments with other benchmarks created similar graphs. At the first time point, the estimated profile is inaccurate due to the lack of knowledge of initial temperatures. In succeeding time points, the estimated temperature profile more closely matches the actual profile. At the 3^{rd} second, a close temperature profile match is achieved. It should be noted that while an *absolute* temperature match is not achieved, a *relative* match across the sensors is provided. In Section IV, this systematic drift is offset by adding constant values to temperatures estimated from sensor readings.

2) Temporal Evolution of Temperature Estimation: Fig. 6(a) shows how the estimated temperature progresses over time for three sensors (integer scheduler, ALU and floating point scheduler). Other sensors show similar trends. Since the initial temperatures are randomly chosen around 55^oC for all

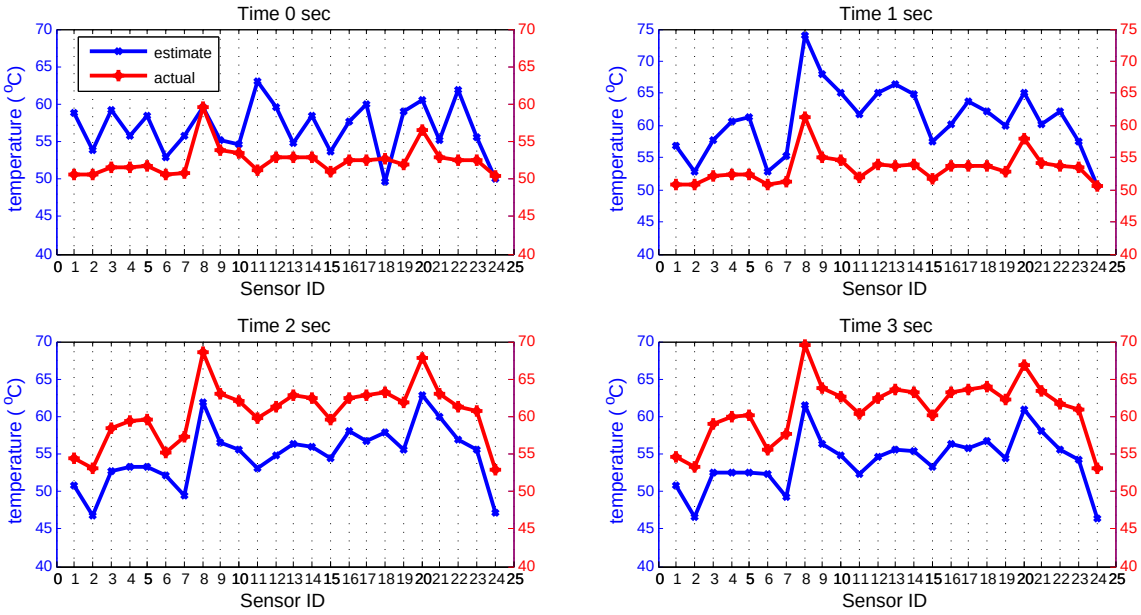

Fig. 5. The estimated and actual processor temperature profile at four time instances for the SPLASH-2 *ocean* benchmark. The estimated thermal profile exhibits a similar shape as the actual profile but with an offset. Each of the 24 thermal sensors in the processor are represented on the horizontal axis for the time instance. The numbered components of AMD processor can be found in the floorplan of Fig. 10. The estimates track the actual temperature with a fixed offset per sensor. This offset $\mathbf{w}$ can be determined (Section IV-B) and compensated.

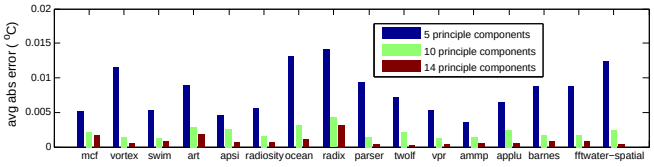

Fig. 7. Prediction accuracy comparison for different numbers of principal components used in the model.

sensors, the estimation mainly reflects heat diffusion during the first 3 seconds. Over time, the temperatures of these three components are corrected to match their approximate relative values (the floating point scheduler is the hottest and ALU is the coolest). Fig. 6(b) shows the correlation coefficient between these two values over time.

To evaluate accuracy, the effect of limiting the number of performance counters used to generate thermal estimates is also considered. Table IV indicates the average absolute error over all sensors and benchmarks for different numbers of performance counters used to generate estimates. Fourteen of the counters provide little benefit in terms of absolute error.

TABLE IV
AVERAGE ABSOLUTE ERROR OF TEMPERATURE ESTIMATION FOR ALL SENSORS AND BENCHMARKS IF THE NUMBER OF PERFORMANCE COUNTERS IS LIMITED TO SPECIFIC QUANTITIES

No. counters	Abs. error (^oC)
5	0.5000
10	0.0164
15	0.0031
20	0.0009
34	0.0009

E. Principal Components of Performance Counter Vectors

In Section III-A it was shown that combinations of performance counter changes and thermal gradients can be combined to estimate temperature changes. However, it has previously been determined [27] that performance counter values are correlated, potentially leading to model instability. For example, a branch miss prediction may lead to a pipeline flush which impacts IPC. To explore the impact of this issue, experiments were performed to replace the P^i values in (3) and (4) with uncorrelated *principal components* [28].

Principal component analysis (PCA) transforms an input vector (in this case u performance counter values) into a new vector set by multiplying the input values with a matrix of the eigenvectors derived from the set, as shown in (12).

$$P'_{1\times u} = P_{1\times u} * C \tag{12}$$

$P_{1\times u}$ is the original vector of u performance counter values collected from the processor and C is a coefficient matrix of eigenvectors determined during model training. $P'_{1\times u}$ is the principal component vector. In many cases, depending on the eigenvalues of the original data set, some of the $P'_{1\times u}$ set may be ignored, leading to a reduced dimension vector $P'_{1\times v}$. Rather than inserting $P_{1\times u}$ performance counters into the linear model in (5), the reduced dimension principal component estimates are inserted instead. Since variables in $P'_{1\times v}$ are orthogonal with each other, the multicollinearity problem is eliminated.

Experimentation showed that the largest 14 principal component estimates correspond to non-zero eigenvalues, but the remaining 20 have eigenvalues close to zero. In general, to maintain maximum accuracy, the number of principal components (e.g. the dimension of v in $P'_{1\times v}$) used in the

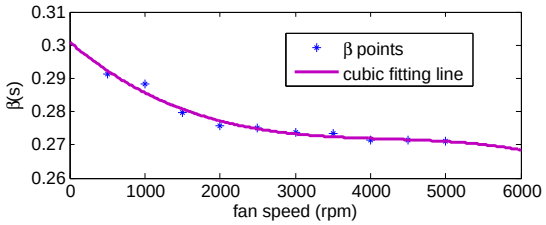

Fig. 8. One β model using cubic fitting for integer module (block 8 in Fig. 10)

model should include the number of non-zero eigenvalues, in this case 14. Fig. 7 shows the thermal estimation error for different numbers of principal components used in the model. As expected, accuracy is improved as the number of principal components is increased from 5 to 14. Principal component count increases beyond this value do not improve accuracy. To assess the benefits of PCA, the experiments described in Section III-C were performed using the fourteen PCA values in place of the thirty-four P^i values as part of the $\mathbf{x^i}$ vector in (5) after model retraining. In all our experiments, the estimated temperature results were nearly identical, indicating the negligible effect of performance counter correlation. As a result, the rest of our reported results use P^i values in (5) rather than PCA values.

F. Dynamic Model for Changing Cooling Conditions

In contemporary computer systems, a variety of cooling technologies (e.g. fans, liquid) are used to efficiently remove heat from the microprocessor and protect it from overheating. Often, the amount of cooling (e.g. fan speed, fluid flow speed) is dynamically adjusted based on a processor's thermal situation. As a result, the β parameters determined through training in Section III-A are only valid for a specific cooling amount. In systems with multiple cooling levels, effectively (5) for thermal sensor i can be restated as:

$$\Delta T^i = \mathbf{x^i} \cdot \boldsymbol{\beta}(s)^i \tag{13}$$

where β(s) values have been determined using the training method described in Section III-B for a specific cooling amount (e.g. fan speed). In this case, β(s) training (Section III-B) is performed at each cooling amount, s. In performing calibration, the appropriate set of β(s) values can be used based on the current cooling amount. The drawback of this method is that it increases storage cost incurred by storing multiple model parameters.

Although this multiple training approach can be effectively used for multiple, discrete cooling amounts, it does not address the issue of a large number of possible cooling amounts. The model to dynamically adapt $\boldsymbol{\beta}$(s) for an s which was not previously trained can be achieved using curve fitting. Fig. 8 shows known β parameters determined through training for integer scheduler as blue stars. The example shown in Fig. 8 exhibits a cubic fit. By building $\boldsymbol{\beta}$(s) models, only a few $\boldsymbol{\beta}$ parameters for specific s cooling amounts must be stored, saving storage space.

IV. Multi-Sensor Collaborative Calibration Algorithms (MSCCA)

Resource-limited thermal sensors, such as ring oscillators, often are affected by noise due to process variation and gradual device wear-out. As a result, their readings may drift away from accurate values. In this section we show that sensor readings can be combined with estimates derived using the performance counter approach from the last section to generate more accurate *corrected* temperature readings. Although it is expected that sensor readings will track corrected readings for long periods of time, if the reading for a specific sensor significantly differs from its corrected readings for a number of samples, the sensor can be recalibrated. In this section, to determine accurate temperature values, estimated temperature values obtained in Section III and readings taken from sensors are merged using a Bayesian inference based algorithm. The corrected temperatures can then be used for thermal calibration.

A. 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.

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{14}$$

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;

$\boldsymbol{\Sigma}_\mathbf{r}$: the covariance matrix of the random vector $\mathbf{r}$;

$\boldsymbol{\Sigma}_\mathbf{e}$: 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.

$$p(\mathbf{t}|\mathbf{r}) = \frac{p(\mathbf{r}|\mathbf{t})p(\mathbf{t})}{p(\mathbf{r})} \tag{15}$$

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 Section III. 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 [16], 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.

$$\boldsymbol{\mu}_t = E(\mathbf{t}|\mathbf{r}) = \int \mathbf{t} \times p(\mathbf{t}|\mathbf{r}) d\mathbf{t} \tag{16}$$

The covariance matrix of the corrected temperature is given as:

$$\boldsymbol{\Sigma}_t = E[(\mathbf{t} - \boldsymbol{\mu}_t)(\mathbf{t} - \boldsymbol{\mu}_t)'] \tag{17}$$

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. $\mathbf{r}|\mathbf{t} \sim \mathcal{N}(\mathbf{t}, \boldsymbol{\Sigma}_\mathbf{r})$. In the Gaussian case, (16) and (17) have closed form representations as follows [29].

$$\boldsymbol{\mu}_\mathbf{t} = \boldsymbol{\mu}_\mathbf{e} + \boldsymbol{\Sigma}_\mathbf{e}(\boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_\mathbf{r})^{-1}(\mathbf{r} - \boldsymbol{\mu}_\mathbf{e}) \tag{18}$$

$$\boldsymbol{\Sigma}_\mathbf{t} = \boldsymbol{\Sigma}_\mathbf{e} - \boldsymbol{\Sigma}_\mathbf{e}(\boldsymbol{\Sigma}_\mathbf{e} + \boldsymbol{\Sigma}_\mathbf{r})^{-1}\boldsymbol{\Sigma}'_\mathbf{e} \tag{19}$$

Thus, the expected actual temperature given $\mathbf{r}$, ($\mu_\mathbf{t}$), and its covariance ($\boldsymbol{\Sigma}_\mathbf{t}$) can be determined directly from sensor readings and estimated temperature values from performance counters.

B. MSCCA Algorithm

The goal of the MSCCA algorithm is to determine the corrected temperature ($\mu_\mathbf{t}$) and covariance ($\boldsymbol{\Sigma}_\mathbf{t}$) for each temperature sensor once per l samples. As seen in Algorithm 1, during each of l samples, temperature sensor readings $\mathbf{r}$ and performance counter values are read. For the sample, the sensor readings from all temperature sensors form a row in an $\mathbf{R}$ matrix (line 8). Additionally, the performance counter values are converted to estimated temperature changes for each sensor using (5) (line 5). These temperature changes are added to the estimated temperatures from the previous sample (line 6) and the results for each sensor is stored in an $\mathbf{E}$ matrix (line 7).

After processing l samples, corrected temperature values for each temperature sensor are determined (line 14) using (18) and (19). Vectors $\mathbf{r}$ and $\mu_\mathbf{e}$ used in the corrected temperature calculation are determined from the columnwise mean of the $\mathbf{E}$ and $\mathbf{R}$ matrices (lines 11 and 12). As noted in Section III and shown in Fig. 5, the use of performance counters to estimate temperature shows a strong relative match, although an absolute offset for the actual temperature is often present. To address this issue, a per-sensor offset value $\mathbf{w}$ is added to each $\mathbf{r}$ reading. Although we found that $\mathbf{w}$ values are constant for each sensor across time and across benchmarks, the values are recalculated in the algorithm for consistency.

TABLE V
OPERATIONS REQUIRED BY MSCCA AND KALMAN FILTERING FOR p SETS OF SAMPLE READINGS FOR 24 THERMAL SENSORS

Operation	Estimation	MSCCA Approach		KF Approach	
		corr.	total	corr.	total
scalar add.	34p×24	$\frac{p}{l}(444l - 48)$	$\frac{p}{l}(1404l - 48)$	0	$816p$
scalar mult.	34p×24	$\frac{p}{l}(300l + 48)$	$\frac{p}{l}(1260l + 48)$	0	$816p$
mat. add.	0	$\frac{p}{l}$	$\frac{p}{l}$	$2p$	$2p$
mat. mult.	0	$\frac{p}{l}$	$\frac{p}{l}$	$10p$	$10p$
mat. -vect. mult.	0	$\frac{p}{l}$	$\frac{p}{l}$	$3p$	$3p$
mat. inv.	0	$\frac{p}{l}$	$\frac{p}{l}$	p	p
vector add.	0	$\frac{p}{l}$	$\frac{p}{l}$	$3p$	$3p$

In our experimentation, calculation was performed over p total samples with l samples per invocation. A total of $\frac{p}{l}$ invocations are used for the p sample set. As shown in Fig. 1, temperature change estimation requires an initial temperature profile of the silicon die which may not be available at runtime. For initialization of the estimation approach, it is possible to assign an arbitrary temperature to each thermal sensor or to use thermal sensor readings as initial temperatures.

Algorithm 1 Multi-Sensor Collaborative Calibration Algorithm – MSCCA

1: Initialize $\boldsymbol{w} \leftarrow \mathbf{0}$.
2: Initialize temperature profile
3: **while** Invocation count $\leq \frac{p}{l}$ **do**
4: **for** $i = 0; i < l; i$++ **do**
5: Estimate Δ temperatures determined from performance counters using (5)
6: Add Δ temperatures to previous corrected temperatures and get updated temperature profile.
7: Store the updated temperatures as a row in $\mathbf{E}$ matrix.
8: Store sensor readings as a row in $\mathbf{R}$ matrix.
9: **end for**
10: Adjust $\mathbf{R}$ matrix by adding offset $\boldsymbol{w}$ to each row.
11: The vector $\mathbf{r}$ is the columnwise mean of $\mathbf{R}$.
12: The vector $\boldsymbol{\mu}_e$ is the columnwise mean of $\mathbf{E}$.
13: Calculate the covariance matrices $\boldsymbol{\Sigma}_\mathbf{r}$ and $\boldsymbol{\Sigma}_\mathbf{e}$.
14: Perform Bayesian inference using Equations (18) and (19), and get the corrected temperature $\boldsymbol{\mu}_\mathbf{t}$.
15: $\boldsymbol{w} \leftarrow \boldsymbol{\mu}_\mathbf{t}$ - $\mathbf{r}$.
16: **end while**

C. Computational Cost Evaluation

This section analyzes the computational complexity of the MSCCA approach and compares it with the complexity of using Kalman filtering to generate corrected temperatures for thermal sensors. Although a full discussion of the KF algorithm for temperature estimation can be found in [18], we provide a brief overview of the required operations here.

The KF approach requires two estimation steps to convert performance counter values to estimated temperature. First, the power consumption of individual functional units is determined using a linear set of equations which have been determined via linear regression [17]. Per-functional unit power values are then converted to estimated temperature via a second set of linear equations [18] whose derivation require the difficult approximation of thermal resistance and capacitance for on-chip functional units. To develop corrected temperature values from estimates and sensor readings, KF then uses cross correlation with previously-determined noise values to merge the estimates and readings together. Unlike our approach, where corrected temperatures are generated every l samples, KF requires corrected temperature evaluation for every sample, a significant time penalty. Thus, our approach has two significant practical benefits versus KF:

- Estimated temperatures are determined directly from performance counter values rather than requiring power as an intermediate value. The elimination of power as a transition metric also eliminates the need for complicated thermal resistance and capacitance calculation.
- MSCCA requires many fewer operations and can be performed less frequently reducing run time.

.

The computational cost for MSCCA and KF approaches (not considering model training which takes place only once at design time) can be broken down into two parts: temperature (or power) estimation and temperature correction. The computational cost of the power estimation for KF and temperature estimation for MSCCA is the time required to calculate a linear combination of scaled performance counter values. As a result, the estimation complexity is $O(np)$, where n is the number of performance counters and p is the number of sample sets. The *estimation* column in Table V shows the number of operations needed to perform this estimation (thermal for MSCCA and power for KF). There are n = 34 performance counters included in our linear regression model, so we specify complexity in terms of this value.

The MSCCA approach stores samples and performs Bayesian estimation once per l time steps. Table V shows the number of operations performed for p sets of readings. For the MSCCA, 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 invocation. 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 $150,000p$ additions and $150,000p$ multiplications required for the KF method. In our method, the numbers of additions and multiplications are about $\frac{p}{l}(1404l + 14,000)$ and $\frac{p}{l}(1260l + 14,000)$.

In contrast, KF-based algorithms predict and update the temperature for *each* set of sample readings, resulting in more matrix operations. In Table V, the $34p$ scalar operations for KF represent the operations to convert power estimates to temperature estimates for a single sensor. The remaining operations represent merging computations for temperature estimates and temperature sensor readings.

D. Memory Cost Evaluation

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. In general, the KF approach does not require storage for the power estimates and sensor reading samples. At each time step, the thermal sensor samples (sensor readings) and temperature estimates determined from power estimations are used to update temperatures, and then these samples are thrown away. The MSCCA approach must store thermal sensor samples $\mathbf{R}$ and temperature estimates $\mathbf{E}$ for l samples in memory until the next MSCCA evaluation. As a result, the memory complexity for the KF approach is $O(1)$ and for MSCCA is $O(ml)$. In our experiments, l is several hundred and m = 24 sensors are used. So the memory storage of samples is around several kilobytes.

E. Implementation Issues

The use of thermal calibration raises concerns about overburdening the hardware and operating system of the target processor. However, the nature of our calibration approach and recent trends in on-chip monitoring for microprocessors lessen this concern. In general, thermal sensor calibration is expected to be performed once every few seconds, rather than milliseconds. In Section VI, it is shown that algorithm execution time is on the order of tens of milliseconds for evaluation that is performed every ten seconds. This overhead limits the operating system and processor-level power and temperature impact of the algorithm itself.

Independent of this overhead limit, recent trends indicate that microprocessors increasingly include *dedicated* circuitry to perform monitoring and monitor data processing which is separate from the main OS/processor compute platform. For example, IBM EnergyScale [2] uses temperature and critical path monitors along with a microcontroller for sensor data processing. Intel's Active Management Technology provides a separate on-chip communications channel and controller to monitor device operation and control system responses at the operating system level. Often, these monitoring and monitor data processing infrastructures can be quite small compared to the main processing infrastructure (e.g. 0.2% of overall processor area [30]), limiting system performance impact. These effects can be weighed against the benefits of a more accurate DTM approach due to improved thermal sensor calibration.

V. Infrastructure and Experimental Approach

In this work, a simulation-based method is employed for data collection, model construction and verification. Our dynamic sensor calibration strategies are verified with the data from simulation. In this section, we describe the two simulators used by this work and other experimental infrastructure.

A. Architectural Simulator

We use the SESC simulator [23] as the infrastructure for collecting system statistics. SESC is a cycle-accurate simulator

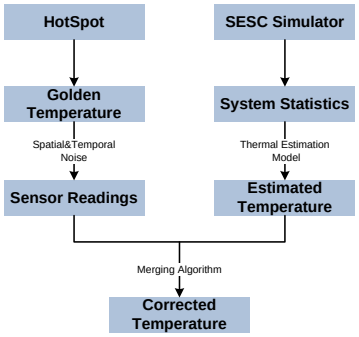

Fig. 9. Sensor data and thermal estimation merging scheme. Sensor readings are artificially generated shown in the left flow; Estimated temperature from performance counter are obtained by the right flow.

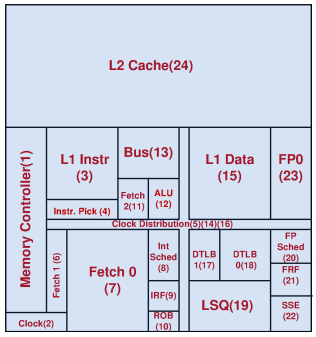

Fig. 10. Floorplan of the Athlon 64 processor [31]

which models a full out-of-order pipeline with branch prediction, caches, buses, and other components of a modern processor. It can also report power traces of system components which are used for thermal simulation. The simulator was modified to support the on-the-fly dumping of performance counter recordings which are synchronized with power traces.

The SESC simulator provides abundant system statistics for architectural analysis. Table I lists a subset of these statistics. Some simulation-related metrics, e.g. simulation speed, are not used in our strategy since it would not be available to a typical many-core user at run-time. 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. 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.

During linear model training for β parameters and for model verification, SESC is used to record the power trace for applications. This information is used by HotSpot [24], a thermal simulator, to determine *actual* temperatures that can be used for training or for comparisons versus thermal estimates to verify our approach (Section V-B). However, since only dynamic power consumption is reported by the simulator and static leakage power accounts for a non-negligible part of total power dissipation for submicron technology nodes (about 40% for our chosen node of 45 nm), we add a static power estimate to the SESC power estimate for each functional unit. First, a dynamic power trace of all function units for a specific application is generated. A percentage of processor dynamic power (40% based on the prediction for 45 nm) is used to estimate static power and a portion of this power is added to the dynamic power trace for each functional unit (proportional to area). To account for the effects of temperature on static power, the power trace is fed to HotSpot and thermal simulation is performed. The static power for each functional unit is then adjusted using thermal dependency linearization [32]. A combination of the adjusted static power and the dynamic power is then used for model training and verification using HotSpot. We have found that the effects of temperature-dependent static power are small over the temperature change range considered.

B. Thermal Simulator

The HotSpot simulation tool, which takes power traces from SESC, target processor geometry and material parameters as inputs, is used to generate accurate "golden" temperatures. As mentioned in the previous subsection, it is assumed that the HotSpot generated temperature values are the *actual* temperatures considering the sophisticated thermal diffusion model implemented by HotSpot (Fig. 9). An AMD Athlon64 processor is used to assess our approach. The floorplan of AMD Athlon64 processor is shown in Fig. 10. The processor includes 24 functional blocks, each of which is labeled in the figure. Each block contains a thermal sensor. According the processor specification of AMD Athlon 64 fabricated under 130 nm SOI technology, the reported die size is 193 mm^2 [33]. After technology scaling, the area of the processor is estimated to be 24 mm^2 in 45 nm technology. The frequency of the processor is configured at 1 GHz in simulation and the overall initial temperature of the processor is set to 50^oC.

C. Spatial and Temporal Noise

Variations in sensor accuracy across temperature sensors on the die (spatial noise) is mainly caused by process variation which is relatively static, so we consider spatial noise to be constant for short time periods (several hours). Unlike spatial noise, variations in a specific sensor's accuracy over time (temporal noise) is caused by environmental effects like voltage and ambient temperature fluctuation, so its value varies for each temperature sample.

VI. Results

For training and verification of our new calibration approach and for comparison to KF, we use the applications listed in Table VI from the SPEC2000 and SPLASH-2 benchmark suites. SPEC2000 is an industry-standardized CPU-intensive benchmark suite which include both integer and floating point applications. To diversify the test benchmarks, we mixed SPEC2000 and SPLASH-2 in the same test sets. These benchmarks were randomly divided into four sets: Set I, Set II, Set III and Set IV. Our thermal estimation model (β values) was trained using benchmark sets I and II. Our models and algorithms are verified with the remaining sets.

TABLE VI
BENCHMARKS USED IN EXPERIMENTAL EVALUATION

Suite Name	Set I	Set II	Set III	Set IV
SPEC2000 CINT	*bzip2, crafty, gzip*	*gap, gcc*	*mcf vortex*	*parser, twolf, vpr*
SPEC2000 CFP	*equake, mgrid*	*mesa, sixtrack, wupwise*	*swim, art, apsi*	*ammp, applu*
SPLASH2	*volrend, cholesky, raytrace*	*lu, water-nsquared, fmm*	*radiosity, ocean, radix*	*barnes, fft, water-spatial*

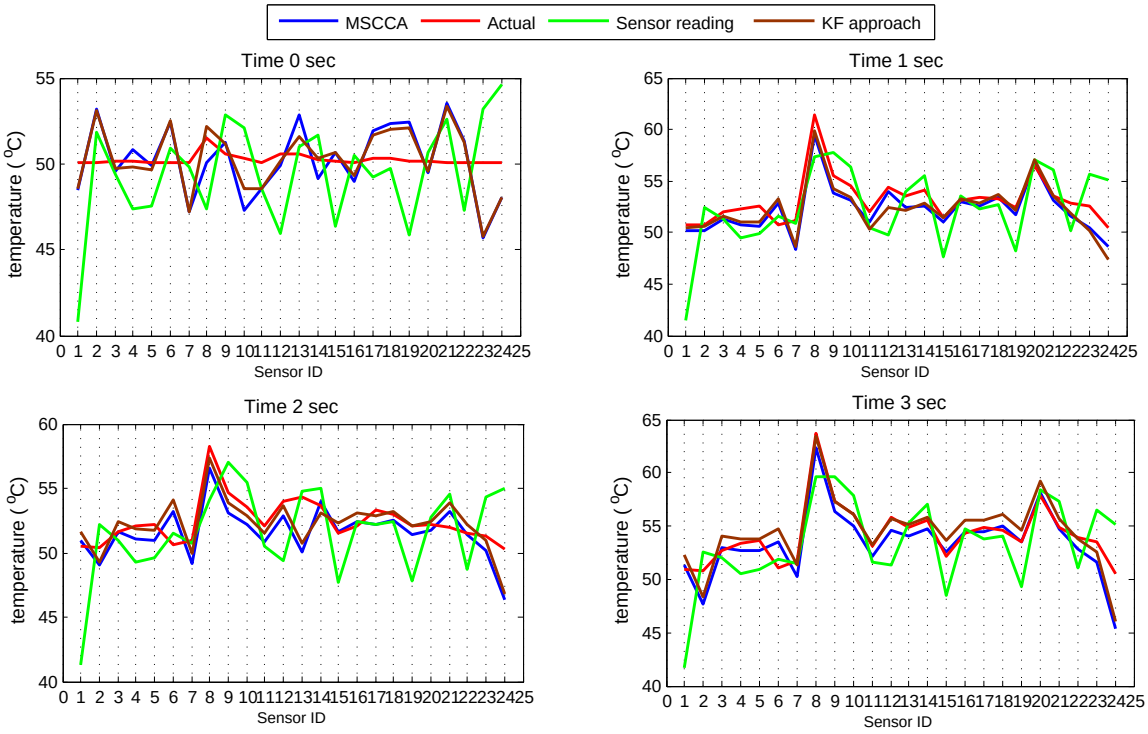

Fig. 11. The estimated and actual processor temperature profile at four time instances. Each of the 24 thermal sensors in the processor (one per functional component) are represented on the horizontal axis for the time instance. The numbered components of AMD processor can be found in the floorplan of Fig. 10. The data was collected through simulation by running *bzip2* on SESC. Other benchmarks show similar convergence rates and temperature results.

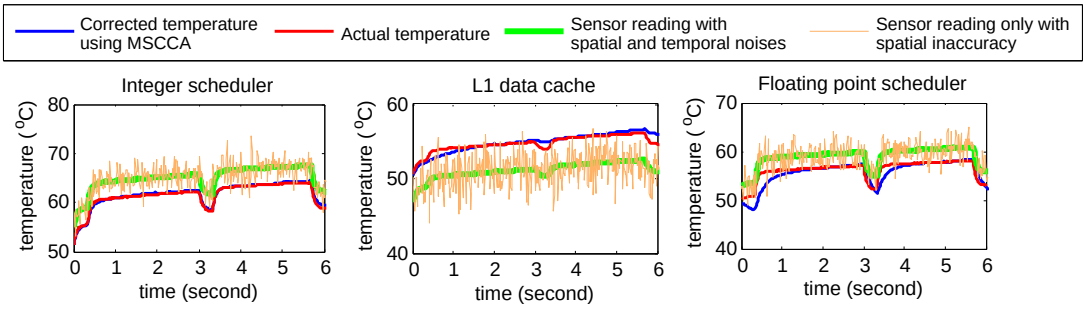

Fig. 12. Temperature tracking over 6 seconds for thermal sensor 8 (integer scheduler), 15 (L1 data cache) and 20 (floating point scheduler) in Fig. 10 running the *twolf* benchmark.

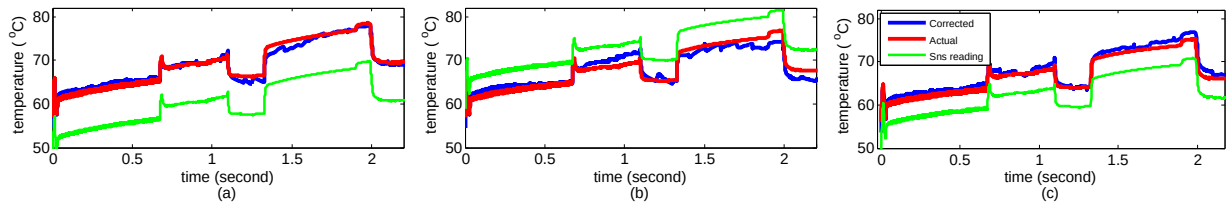

Fig. 13. (a) Temperature tracking with 800 rpm fan speed. (b) Temperature tracking with 2800 rpm fan speed. (c) Temperature tracking with 4800 rpm fan speed. All three experiments use the benchmark *volrend*.

A. Effectiveness Verification

In a first series of experiments, the thermal profiles for the AMD Athlon 64 were determined using Algorithm 1. For these experiments, the standard deviation of temporal and spatial noises were both set to 4^oC and 6000 total time instances of readings were processed. MSCCA uses l=100 time instances per invocation. In Fig. 11, we demonstrate the thermal profile of the AMD Athlon 64 processor for the *bzip2* benchmark. The horizontal axis represents thermal sensors for each functional block in Fig. 10. In the figure, the actual temperature, sensor readings (only spatial noise is shown for clarity but the experiment is performed with both spatial and temporal noises), corrected temperature from the KF-based implementation, and corrected temperature from MSCCA using thermal estimates from performance counters are plotted. It is apparent 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.

Although corrected temperatures initially differ from actual temperatures due to incorrect initial estimates of temperature, the corrected temperatures determined by MSCCA quickly converge. In both the MSCCA and KF cases, the thermal profile is recovered after synthesizing two data sources (the estimated temperature and sensor readings in the MSCCA case, the statistical characteristics of the power dissipation and sensor readings in the KF case). The MSCCA case has the benefit of faster calculation (contrasted in Section VI-E) and a much simpler model training process (no power-to-temperature model needed).

B. Temperature Tracking Using MSCCA

Fig. 12 shows the temperature tracking results for three thermal sensors: Sensor 8 (integer scheduler), Sensor 15 (L1 data cache) and Sensor 20 (floating point scheduler). The results from other sensors are similar. For each sensor, four curves are plotted in the figure: the actual temperature, MSCCA corrected temperature, noisy sensor readings and noisy sensor reading with temporal noise pruned out. Since we assume that the spatial noise does not change in a short time period, the green curve has a constant offset from red curve. The simulation lasts for 6 seconds and the initial temperatures for all sensors are randomly generated.

Although the actual initial temperatures for the three sensors are not 50^oC, MSCCA estimation results were generated using this initial value. Estimated temperature values converge to the actual temperature over time. The figure indicates that sensors 8 and 20 show good calibration accuracy since the estimation curves are closer to the actual curves than the sensor reading curves.

Fig. 13 shows temperature tracking results with different fan speeds, 800, 2800 and 4800 rpm respectively. A different set of β parameters are used for each fan speed, as discussed in Section III-F. The thermal model can effectively track the actual temperature values for all fan speed values, as expected.

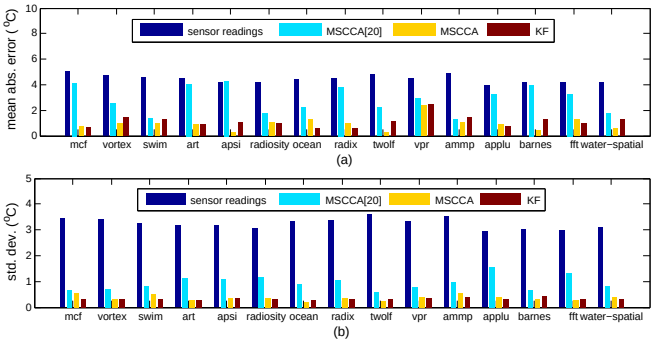

Fig. 14. (a) The average absolute error for sensor readings, temperatures generated with MSCCA, and with KF (b) The standard deviation of errors for sensor readings, temperatures generated with MSCCA, and KF approaches.

TABLE VII
THE STANDARD DEVIATION OF THE ERROR FOR THE CORRECTED TEMPERATURES OVER 10,000 TIME INSTANCES FOR INCREASING SENSOR ERROR

Time instances per invocation l	Std dev of sensor error ^oC			
	2	4	6	8
100	0.1059	0.1575	0.2047	0.2798
200	0.1137	0.1663	0.2159	0.2935
300	0.1203	0.1711	0.2241	0.3040
400	0.1276	0.1732	0.2284	0.3093
500	0.1311	0.1819	0.2372	0.3187

C. Estimation Error Comparison

The experiments in the previous subsection qualitatively show the effectiveness of dynamic sensor calibration using data fusion. In this section, the error of the MSCCA approach is quantitatively evaluated. Fig. 14 shows the average absolute error and the standard deviation of the errors for original sensor readings, MSCCA in [20], MSCCA in this paper, and the KF approach. In [20], absolute T rather than ΔT values were determined using (5). The MSCCA [20] results in Fig. 14 were determined using the same training set as the new MSCCA algorithm. The spatial noise added to sensor readings is Gaussian with standard deviation 6^oC and the temporal noise is Gaussian with 4^oC standard deviation.

In Fig. 14(a), the average absolute error is reduced by 5^oC (from 6^oC to 1.2^oC) with respect to the original sensor readings. In Fig. 14(b), the standard deviation of the error is reduced by a factor of 10 (from 3^oC to 0.2^oC) from the original sensor readings with limited computational effort.

D. Impact of Sensor Reading Noise

The standard deviation of the errors of the corrected temperature increases as the noise of the sensor readings becomes larger. The experiments in Section VI-C were repeated, this time with varying amounts of noise in the sensor readings. Experiments of 10,000 time instances each were performed. To better evaluate the effect of noise, one set of 10,000 random noise values was determined for each noise amount (e.g. each column in Table VII). These values were added to read values and the results are used for comparison across configurations. The table shows the standard deviations of 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.

E. Run Time Comparison

Table VIII shows the total run time including both temperature estimation and correction for both MSCCA and KF approaches. The total number of samples is 10K, collected in 10 seconds. The total run time per sampling interval for MSCCA is <0.004 sec for MSCCA and about 0.2 sec for the KF-based approach. As seen in the table, the run time of MSCCA decreases as the number of time instances per MSCCA invocation increases and it remains constant for the KF approach since corrected temperature calculation is performed for each sample.

TABLE VIII
RUN TIME COMPARISON (IN SECONDS) BETWEEN MSCCA AND KF APPROACHES FOR **10000** TIME INSTANCES

Instances/invocation	100	200	400	500	1000
MSCCA run time	0.0387	0.0301	0.0242	0.0228	0.0169
KF run time	1.9076	1.912	1.9289	1.9133	1.8946

VII. CONCLUSION

In this research, an on-line model is presented to estimate temperatures of multiple sensor locations on the silicon die using information from performance counters. By merging noisy sensor readings and these estimated temperatures using our new MSCCA algorithm, corrected temperature readings for thermal sensors are achieved. Our strategy is evaluated using SPLASH-2 and SPEC2000 benchmarks suites. Results show that the strategy can effectively recalibrate sensor readings in response to inaccuracies caused by process variation and environmental noise. The average absolute error of the corrected sensor temperature readings is $< 1.5^oC$ and the standard deviation of error is less than $< 0.5^oC$ for tested benchmarks. Our overall estimation and correction run time is significantly reduced versus Kalman filtering technique (at least $50\times$ faster) to make our strategy favorable for real time implementation. Future work includes the development of an improved approach to integrate temperature-dependent leakage into our temperature sensor calibration approach.

REFERENCES

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

[2] M. Floyd, et al., "Adaptive energy-management features of the IBM POWER7 chip," *IBM Journal of Research and Development*, vol. 55, no. 3, pp. 8:1–8:18, 2011.

[3] E. Rotem, et al., "Power management architecture of the 2nd generation Intel core microarchitecture, formerly codenamed Sandy Bridge," in *Hot Chips*, August 2011.

[4] 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*, May 2009, pp. 487–492.

[5] H. Hamann, et al., "Hotspot-limited microprocessors: Direct temperature and power distribution measurements," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 1, pp. 56–65, Jan. 2007.

[6] Y. Zhang and A. Srivastava, "Adaptive and autonomous thermal tracking for high performance computing systems," in *ACM/IEEE Design Automation Conference*, Jun. 2010, pp. 68 –73.

[7] C. Lian, et al., "Development of a flexible chip infrared (IR) thermal imaging system for product qualification," in *Semiconductor Thermal Measurement and Management Symposium*, 2012, pp. 337–343.

[8] S. Reda, "Thermal and power characterization of real computing devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 76 –87, June 2011.

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

[10] F. Liu, "A general framework for spatial correlation modeling in VLSI design," in *IEEE/ACM Design Automation Conference*, Jun. 2007.

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

[12] J. Ranieri, et al., "Eigenmaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors," in *IEEE/ACM Design Automation Conference*, 2012, pp. 636–641.

[13] H. Zhou, et al., "An information-theoretic framework for optimal temperature sensor allocation and full-chip thermal monitoring," in *IEEE/ACM Design Automation Conference*, 2012, pp. 642–647.

[14] A. Kumar, et al., "System-level dynamic thermal management for high-performance microprocessors," *IEEE Trans. on CAD*, vol. 27, no. 1, pp. 96 –108, Jan. 2008.

[15] K.-J. Lee and K. Skadron, "Using performance counters for runtime temperature sensing in high-performance processors," in *IEEE Int'l Parallel and Distributed Processing Symp.*, April 2005, p. 232.

[16] S. Sharifi and T. Rosing, "Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management," *IEEE Trans. on CAD*, vol. 29, no. 10, pp. 1586–1599, Oct. 2010.

[17] M. Powell, et al., "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters," in *IEEE Int'l Symp. on High Performance Computer Arch.*, Feb. 2009, pp. 289–300.

[18] Y. Zhang and A. Srivastava, "Accurate temperature estimation using noisy thermal sensors for Gaussian and non-Gaussian cases," *IEEE Transactions on VLSI*, vol. 19, no. 9, pp. 1617 –1626, Sept. 2011.

[19] ——, "Accurate temperature estimation using noisy thermal sensors," in *ACM/IEEE Design Automation Conference*, Jul. 2009, pp. 472 –477.

[20] S. Lu, R. Tessier, and W. Burleson, "Collaborative calibration of on-chip thermal sensors using performance counters," in *Proc. of the IEEE/ACM Int'l Conf. on CAD*, Nov. 2012, pp. 15–22.

[21] 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*, Jul. 2006, pp. 554 –557.

[22] S. Woo, at al., "The SPLASH-2 programs: characterization and methodological considerations," in *Proc. Int'l Symp. on Computer Arch.*, 1995.

[23] Jose Renau, et al., "SESC simulator," January 2005, http://sesc.sourceforge.net.

[24] "Hotspot," http://lava.cs.virginia.edu/HotSpot/.

[25] D. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, May 1992.

[26] R. M. Neal, *Bayesian Learning for Neural Networks*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996.

[27] R. Cochran and S. Reda, "Thermal prediction and adaptive control through workload phase detection," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 1, pp. 7:1–7:19, Jan. 2013.

[28] R. Johnson and D. Wichern, *Applied Multivariate Statistical Analysis*. Prentice Hall, 2007.

[29] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors," in *Int'l Conf. on Wireless Sensor Networks and Apps.*, Sep. 2003.

[30] J. Zhao, et al., "A dedicated monitoring infrastructure for multicore processors," *IEEE Transcations on VLSI*, vol. 19, no. 6, pp. 1011–1022, Jun. 2011.

[31] F. J. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau, "Power model validation through thermal measurements," in *International Symposium on Computer Architecture*, Jun. 2007, pp. 302–311.

[32] Y. Liu, R. Dick, L. Shang, and H. Yang, "Accurate temperature-dependent integrated circuit leakage power estimation is easy," in *Design, Automation and Test in Europe*, 2007.

[33] "Athlon 64 Clawhammer," http://www.cpu-world.com/.