Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management

Shervin Sharifi *Student Member, IEEE,* and Tajana Šimunić Rosing, *Member, IEEE*

***Abstract*—Dynamic thermal management techniques require accurate runtime temperature information in order to operate effectively and efficiently. In this paper, we propose two novel solutions for accurate sensing of on-chip temperature. Our first technique is used at design time for sensor allocation and placement to minimize the number of sensors while maintaining the desired accuracy. The experimental results show that this technique can improve the efficiency and accuracy of sensor allocation and placement compared to previous work and can reduce the number of required thermal sensors by about 16% on average. Secondly, we propose indirect temperature sensing to accurately estimate the temperature at arbitrary locations on the die based on the noisy temperature readings from a limited number of sensors which are located further away from the locations of interest. Our runtime technique for temperature estimation reduces the standard deviation and maximum value of temperature estimation errors by an order of magnitude.**

***Index Terms*—Multiprocessor SoC, sensor placement, temperature difference, thermal management, thermal sensor.**

I. Introduction

HIGH TEMPERATURES and temperature variations cause degraded reliability, slower devices, increasing resistances and leakage power, and so on [2]. More than 50% of all integrated circuit failures are related to thermal issues [2]. Due to such problems, high temperatures caused by ever increasing miniaturization and increased power densities in new generations of very large scale integrated circuits require that thermal considerations be taken into account during design, manufacturing, test, and at runtime [2]. Dynamic thermal management (DTM) techniques adapt runtime behavior of the chip to achieve the highest performance under thermal constraints [3]. One of the most important aspects of DTM is to capture the runtime variations in the temperature caused by power consumption variations due to workload changes. This is necessary for accurate and timely responses to thermal emergencies.

Accuracy of thermal measurements directly affects the efficiency of thermal management as well as the performance of the CPU [8]. Temperature estimations lower or higher than the actual temperature may cause late or early activation of DTM techniques. Late activation of DTM can result in degraded reliability since the temperature may exceed the designated thresholds. Early activation of DTM can have significant impact on performance, especially in the case of response mechanisms with high invocation times and overhead (e.g., dynamic voltage and frequency scaling). DTM techniques usually rely on temperature information obtained from on-chip thermal sensors or on online thermal models which estimate the temperature based on power consumption of different units. On-chip thermal sensors are the most popular means of obtaining runtime temperature information required for DTM. Different numbers of sensors are deployed on various modern processors. Cell processor with 9 cores contains 11 thermal sensors [38] and Dunnington Xeon processor with 6 cores contains two thermal sensors per core [39]. An analog on-chip thermal sensor usually consists of a temperature-sensing diode, a calibrated reference current source, and a current comparator. Components other than the thermal diode (e.g., current source) must be placed as far as possible to the hotspot due to their temperature sensitivity. Moreover, the sensed temperature values must be routed to where they are used. Routing overheads associated with these can significantly contribute to the overall cost [34].

One of the major problems in direct use of on-chip temperature sensors is the sensor imprecision and noise [5], [6]. There are several factors that cause inaccuracy in temperature measurements. The sensor placement error is one of the most important sources of inaccuracy in values obtained from thermal sensors. Typically, sensors cannot be placed right at the locations they are supposed to monitor, since hotspots usually happen at high-performance and high-density areas where silicon is at a premium. This causes a disparity between the actual temperature at the location of interest and the sensor. Increasing the number of sensors can resolve this issue, but the cost of adding a large number of sensors is prohibitive. Moreover, even without considering the cost of sensors, various other limitations such as the need for more

Manuscript received September 27, 2009; revised January 11, 2010; accepted May 3, 2010. Date of current version September 22, 2010. This work was funded by the NSF Project GreenLight, under Grant 0821155, NSF Grants 0916127 and 1029783, SRC Grant P11816, MuSyC, DARPA, UC Micro, IBM, Texas Instruments, Oracle, Qualcomm, Cisco, and UCSD Center for Networked Systems. This paper was recommended by Associate Editor N. Chang.

The authors are with the Department of Computer Science and Engineering, University of California, San Diego, CA 92093 USA (e-mail: shervin@ucsd.edu; tajana@ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2061310

channels for routing and input/output may not allow placement of thermal sensors right on the locations of interest.

There are also other factors affecting the accuracy of sensor readings. Variations in process parameters introduced during manufacturing can result in sensor reading inaccuracies (e.g., threshold voltage variation on the die) [6]. Errors are also introduced in the process of analog to digital conversion due to quantization, and due to limitations of design and technology. Changes in power supply voltage can affect sensor readings. Finally, the statistical characteristics of sensor inaccuracy change during the lifetime of the chip [7].

Recent work reports thermal sensor accuracy of around $\pm 1\,^{\circ}\mathrm{C}$ [34], however, this is achievable only through accurate calibration [35]. Many microprocessors use un-calibrated thermal sensors [35]. This is mainly due to the high cost and overhead associated with the thermal sensor calibration, particularly in the systems featuring multiple sensors. The calibration is usually done at test time and thus incurs overhead in design and test cost and silicon area [6]. It requires pre-heating and testing the sensors to detect various errors. Once these errors and sources of inaccuracy are known, the sensing unit is calibrated using A/D converters and look-up tables. Even well calibrated sensors are not capable of accurately reporting the actual hot spot temperature on the die due to their location, as explained before. Even when the average sensor error is low, the sensor might deliver single readings that are quite different from the actual temperature. If the readings from the sensors are directly used with complete trust, such variations may cause problems such as performance degradation due to early activation of DTM, or reliability degradation due to its late activation.

In this paper, we propose techniques which address different sources of inaccuracies in sensing temperature using thermal sensors. The first technique we propose in this paper is a *design time* technique which addresses the problem of efficient placement of on-chip temperature sensors on the die while guaranteeing the desired accuracy at each location of interest. It finds the minimum number of sensors and their locations to cover a number of locations of interest with a maximum acceptable sensor placement error. This technique is based on our analytical model for finding the upper bound of on-chip temperature differences. Our second technique, indirect temperature sensing, is a *runtime* technique to address issues such as unavailability of enough sensors, degradation or failure of existing sensors and dynamic change of hotspot locations. This technique accurately estimates the temperature at different locations on the die using noisy readings obtained from a few available sensors and power estimates of functional units. It also complements our design time technique by allowing a trade-off between hardware cost and computation cost. Now we can use fewer sensors at the cost of slightly increased computation at runtime. Due to its low overhead it can be used for temperature aware scheduling and other online thermal management techniques. Indirect temperature sensing can be activated as needed rather than continuously; for example when the temperature at a unit on the chip is approaching a threshold. In this way, it provides an easy trade off between accuracy and overhead. Finally, it can adapt to changes in measurement noise characteristics, which is very important since mean time to failure of thermal sensors is shorter than that of assets they are supposed to protect. We have implemented and assessed these techniques. Our experimental results show that our sensor placement algorithm results in an average of 16% reduction in number of sensors compared to the previous techniques at no cost. Our indirect temperature sensing technique also shows an order of magnitude reduction in the standard deviation and maximum value of temperature estimation error relative to measured temperature values.

The next section describes the related work. Section III explains the details of our analytical model for upper bound on temperature difference that is then used in Section IV for the proposed sensor placement method. Section V explains our indirect temperature sensing. Experimental results are provided in Section VI and Section VII concludes this paper.

II. Related Work

One of the most widely used models for temperature estimation at micro-architectural level is HotSpot [8], which is based on building a multilayer thermal *RC* network of the given chip. The differential equations used to describe the heat flow have a form similar to that of electrical current. This duality is the basis for the micro-architectural level thermal model of HotSpot which was proposed in [8] and further described in [9] and [1]. Temperature modeling techniques such as HotSpot incur too high-computation cost during runtime. In [11], a technique is proposed which performs runtime thermal simulation based on the observation that the average power consumption of architecture level modules in microprocessors is the major contributor to the variations in the temperature. Therefore, piecewise constant average power inputs can be used to speed up the thermal analysis. Techniques such as those introduced in [11] need to continually perform temperature estimation, thus causing significant overhead. In addition, since these on-chip temperature estimation techniques are not combined with thermal sensor measurements, the estimates can easily deviate from the actual values. Due to these issues, temperature information for DTM is usually obtained from thermal sensors at runtime.

In [36], the authors address the problem of estimating the accurate sensor temperatures given noisy sensor readings. This paper focuses on steady-state temperature and doesn't consider transient temperature changes, which is necessary for dynamic thermal management techniques. Temperature is assumed to depend only on the current power consumption of the functional blocks and dependence of current temperature on previous temperature is ignored. In [37], the authors present a thermal characterization approach which reconstructs the thermal map of the die based on limited sensor measurements using spectral Fourier analysis techniques. This approach is able to reconstruct the thermal map using limited sensor data, but the effect of inaccurate and noisy sensors is not considered. The information about the thermal map is extracted only based on the thermal sensor readings and thermal dynamics of the chip. Power consumption of different cores are not exploited. None of the previously published techniques provide

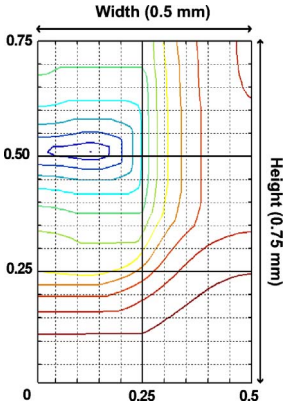


Fig. 1. Contour map of maximum temperature difference to a point of interest.

any results on their execution times to show practicality of these techniques for thermal estimation at runtime.

Different techniques have been proposed for efficient placement of on-chip thermal sensors. These techniques are usually based on identification of the hotspots and placing the minimum number of sensors such that they appropriately cover these hotspots. A sensor placement method is proposed in [21] which is based on the concept of range around a hotspot. This is the maximum distance from the hotspot within which a sensor can be placed while still maintaining the intended accuracy. This technique estimates the maximum temperature difference between a heat source and its surrounding locations based on their distance. The model is based on the assumption that the temperature decays exponentially with this distance from a hotspot. A maximum distance from the hotspot is calculated where temperature difference of all of the points within this distance to the hotspot is less than a maximum acceptable temperature error. Sensors are placed within this distance from the hotspot in order to maintain a desired level of accuracy. Selection of activity factor parameter in this technique is not easy and also depends on the application. Therefore, the results will not be exact and a pessimistic estimation must be used to guarantee the maximum error. Moreover, when calculating the maximum temperature difference to a hotspot, the result depends only on the distance from the hotspot. In other words, it implies that for all of the points at equal distance from the hotspot, the maximum temperature difference to the hotspot is the same, which is not correct. This can be due to the effect of the location and power consumptions of other power sources on the temperature around a hotspot. Fig. 1 shows the contour map of maximum temperature difference relative to a point of interest in a multiprocessor SoC which is used in our experiments and consists of 6 *XScale* cores [16]. This figure clearly shows that the maximum temperature differences around the region of interest are not the same for equidistant points from the hotspot.

In [20], the authors introduce a systematic technique for thermal sensor allocation and placement in microprocessors. This technique identifies an optimal physical location for each sensor such that the sensor's attraction toward steep thermal gradient is maximized. However, this approach does not consider the accuracy of the sensors and does not guarantee the maximum error in the thermal sensor readings.

To the best of our knowledge, no technique has been previously proposed to estimate the temperature difference between various locations on the die. The technique proposed in [21] is a special case of this problem and proposes a model for estimating the temperature at distance d from a heat source. In this paper, we propose an analytical model for estimating an upper bound for the maximum temperature difference between any two points on the die. Usually in order to estimate the maximum temperature differences and variations, extensive simulations are performed. The upper bound provided by our model is not application dependent and does not involve extensive simulations. Other than sensor placement, the proposed model can be used in analyzing the worst case temperature variations on the chip such as analysis of reliability and performance mismatch. Spatial and temporal temperature variations happen as a result of functional and structural differences and differences between computational activities across the chip and also workload variations during the time. Temperature variations as high as 50 °C across the die in a modern microprocessor are reported in [2]. These variations impact reliability and performance of the systems. It is shown in [24] that at moderate temperatures, spatial and temporal temperature gradients determine the device reliability; and to achieve satisfactory reliability, resolving the thermal hotspots alone is not adequate. In [23], a comprehensive framework is proposed for analyzing the effects of temperature and temperature variations on reliability of multi-core systems. Temperature variations may also cause performance mismatch which can lead to performance or functional failures. For example, since wire resistances scale with temperature, difference between temperatures of two regions of the die cause difference between resistances at those two regions which may result in timing issues in interconnects and clock skew problems in clock networks. This makes temperature variations an important factor in clock tree design and optimization [25]. Such issues make analysis of temperature variations and gradients an important issue. Our model which is presented in the subsequent section is a useful tool for analyzing maximum temperature variations across the die.

III. Analytical Model for Upper Bound of On-Chip Temperature Differences

In this section, we present the analytical model which is later used by our proposed sensor placement algorithm. Finding the maximum temperature difference between various locations on the die is an important step during the thermal analysis and design as it enables better placement of temperature sensors and helps in evaluation of reliability issues. The maximum temperature difference under potential workloads can be found by extensive simulations, which would incur significant overhead. For systems whose operation depends on interaction with other systems, even the same workload can result in completely different behavior, thus introducing an even higher overhead for temperature estimation. In contrast, our method provides an upper bound with insignificant overhead.

Our techniques in this paper are based on the same thermal model as used in HotSpot. The thermal network generated by the Hotspot model includes thermal resistors and capacitors.

Temperature can be modeled at the level of a functional block, or the die can be divided into regular grid cells (Fig. 1) to obtain more fine-grained estimates. Given the layout and the thermal characteristics of a chip, it is divided into a grid of r rows and c columns as shown in Fig. 1. The proper size of the grid cells and the number of rows and columns of the grid can be determined by the method proposed in [1]. Our technique works at the granularity of a grid cell, therefore, when we talk about different locations or points of interest; we actually refer to the corresponding grid cell.

The algorithm starts with the evaluation of the effect each power source has on the temperature differences. This can be done by simulation or with analytical methods. Following this step, the maximum temperature difference between pairs of locations is calculated by exploiting the linear time-invariant (LTI) characteristics of the system. Since the thermal resistors and capacitors are linear components, the thermal network can be considered an LTI dynamic system. We exploit the LTI characteristics of this system as a basis for calculating an upper bound for the temperature difference. First, we explain it on a simple case of a single input and single output system, and then extend to the thermal networks with multiple inputs and outputs. If we define the power input to the thermal circuit as $p(t)$, the impulse response of the system as $h(t)$, then the temperature output of the system $f(t)$ can be represented as

$$f(t) = h(t) * p(t) = \int_0^t h(\tau)\, p(t-\tau)\, d\tau. \tag{1}$$

We assume that minimum and maximum power consumed at each functional unit, p^{Min} and p^{Max}, are known, and that the power consumed at a functional unit is always positive

$$0 \le p^{\text{Min}} \le p(t-\tau) \le p^{\text{Max}}. \tag{2}$$

$H+$ and $H-$ are the sets of intervals where the impulse response $h(t)$ takes non-negative and negative values, respectively. Therefore, the temperature output of the system can be represented as

$$\begin{aligned} &f(t) = \textstyle\int_0^t h(\tau)p(t-\tau)\, d\tau = \\ &\int_{H+} |h(\tau)|\; p(t-\tau)\, d\tau - \int_{H-} |h(\tau)|\; p(t-\tau)\, d\tau. \end{aligned} \tag{3}$$

Based on (2), we know that

$$\begin{aligned} &p^{\text{Min}} \int_{H+} |h(\tau)|\; d\tau \;\le \int_{H+} |h(\tau)|\; p(t-\tau)\, d\tau \\ &\le\; p^{\text{Max}} \int_{H+} |h(\tau)|\; d\tau \\ &p^{\text{Min}} \int_{H-} |h(\tau)|\; d\tau \;\le \int_{H-} |h(\tau)|\; p(t-\tau)\, d\tau \\ &\le\; p^{\text{Max}} \int_{H-} |h(\tau)|\; d\tau. \end{aligned} \tag{4}$$

Equation (4) enables us to then derive the bounds on the value of the output f of a single input single output system based on its impulse response. If we assume the bounds to be f^{Min} and f^{Max} ($f^{\text{Min}} \le f(t) \le f^{\text{Max}}$), they are calculated as

1. Initialization
for all grid cells of interest,
Find $h_{j,i}(t)$ (By simulation-based or analytical methods)
2. Upper bound calculation
for each pair of grid cells of interest (a,b)
Find f^{Min} and f^{Max} using impulse responses initialized in step 1

Fig. 2. Algorithm for calculating the upper bounds.

$$\begin{aligned} f^{\text{Min}} &= p^{\text{Min}} \int_{H+} |h(\tau)|\; d\tau \;-\; p^{\text{Max}} \int_{H-} |h(\tau)|\; d\tau = \\ &\quad p^{\text{Min}} \int_{H+} h(\tau)\, d\tau \;+\; p^{\text{Max}} \int_{H-} h(\tau)\, d\tau \\ f^{\text{Max}} &= p^{\text{Max}} \int_{H+} |h(\tau)|\; d\tau \;-\; p^{\text{Min}} \int_{H-} |h(\tau)|\; d\tau = \\ &\quad p^{\text{Max}} \int_{H+} h(\tau)\, d\tau + p^{\text{Min}} \int_{H-} h(\tau)\, d\tau. \end{aligned} \tag{5}$$

We represent a system with m power sources as $p_1(t), \ldots, p_m(t)$, where for each power source maximum and minimum input values are defined as p_i^{Max}, p_i^{Min}. Assuming $h_i(t)$ is the response of the single output to the impulse on input i, the LTI characteristics of the system imply

$$f(t) = \sum_{i=1}^{m} f_i(t) = \sum_{i=1}^{m} h_i(t) * p_i(t). \tag{6}$$

Therefore, the minimum and maximum values of the function are

$$\begin{aligned} f^{\text{Min}} &= \sum_{i=1}^{m} f_i^{\text{Min}} \\ f^{\text{Max}} &= \sum_{i=1}^{m} f_i^{\text{Max}}. \end{aligned} \tag{7}$$

Equation (7) holds for all outputs of the system. The temperature difference between grid cells a and b is represented by $T_d(a, b)$. Its impulse response to input i, $h_{(a,b),i}(t)$, can be calculated as

$$h_{(a,b),i}(t) = h_{a,i}(t) - h_{b,i}(t) \tag{8}$$

where $h_{a,i}(t)$, $h_{b,i}(t)$ are impulse responses of temperature at grid cells a and b, respectively. Calculation of $T_d(a, b)^{\text{Min}}$ and $T_d(a, b)^{\text{Max}}$ for each output requires only the knowledge of ${p_i}^{\text{Max}}$, ${p_i}^{\text{Min}}$ and $h_{(a,b),i}(t)$ as can be seen from (5). The impulse response characteristics of the chip, $h_{(a,b),i}(t)$ can be calculated by simulation or by analytical methods.

The process of finding maximum temperature differences is shown in Fig. 2. Initialization step needs to be done just once, and then its results can be used for the upper bound calculation. To find $h_{j,i}(t)$ during initialization, a step input is applied to power source i while setting all other power sources to 0. Then, the step response at grid cell of interest j is used to calculate the impulse response by differentiation. $h_{j,i}(t)$ can also be calculated by analytical methods of linear systems theory.

After initialization, upper bound is calculated for each pair of grid cells of interest. First, the impulse response of temperature difference between the two points caused by power source i is calculated ($h(t) = h_{a,i}(t) - h_{b,i}(t)$). Then,

based on this impulse response, T_d^{Min} and T_d^{Max} are calculated using (5) and (7).

The calculations can be done only for the pairs of grid cells which are of interest, after completing the simulations for the initialization step (which are done just once). For example, we can use this algorithm to find maximum temperature difference between hotspot a and the set of potential sensor locations around that hotspot $L = \{l_1, l_2, \ldots, l_k\}$. In this case, the grid cell pairs of interest are $(a, l_1), \ldots, (a, l_k)$. As explained before, many constraints, such as routing, can limit the number of potential sensor locations around a hotspot.

The model can also be used to generate the power trace which results in the maximum temperature difference between any two points. Using (5) to maximize the value of *f(t)* at t_0, we know that $p(t_0 - \tau)$ must take the maximum value at the intervals of τ where $h(\tau) = 0$. For example, if $h(\tau)$ is non-negative on interval $t_1 < \tau < t_2$, $p(t)$ must take its maximum value on interval $t_0 - t_2 < t < t_0 - t_1$. Similarly, it can be shown that for the intervals of τ where $h(\tau) < 0$, $p(t_0\text{-}\tau)$ must take its minimum value. These rules allow us to generate the power trace *p(t)* which leads to the maximum temperature difference between two arbitrary points. Doing this for all power sources enables us to detect the configurations and scenarios which lead to maximum temperature variations between different points on the die. This information can also be helpful in augmenting the benchmarks and generating test data for the stress tests.

In the next section, we propose an efficient thermal sensor placement technique which uses our analytical model to minimize the number of thermal sensors while keeping the sensor placement error within acceptable limits.

IV. Thermal Sensor Placement

The objective of our sensor placement technique is to find the minimum number of sensors and their locations such that the temperature reading errors for each point of interest is within the required accuracy. As shown in Fig. 1, the chip is divided into a grid. Whenever we specify a point on the die, we are refering to the corresponding grid cell at that location. Let's suppose $Q = \{q_1, q_2, \ldots, q_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$ are the set of n points of interests and the set of corresponding desired accuracies for these points, respectively. We also define a set of potential sensor points $L = \{l_1, l_2, \ldots, l_k\}$ which consists of all of the grid cells around the hotspots where a temperature sensor can be placed. This set is usually determined by other design considerations such as availability of space for locating the sensor, etc. Also, usually sensors could not be placed inside on-chip memory blocks, routing, and IO.

The objective is to find the minimum set of sensors (and their locations) $S = \{s_1, s_2, \ldots, s_m\}$ such that for each q_i there exists a s_j for which $T(q_i) - T(s_j) < e_i$ always holds. S must be a subset of L.

Earlier, we introduced the concept of observability area which is defined as the area around a point of interest q within which the maximum temperature difference is always less than the maximum tolerable error. Therefore, if a sensor is placed in the observability area of a point of interest, the sensor placement error would be less than the maximum tolerable error. The observable set of a point of interest q is the set of grid cells in L which completely fall in its observable area. We can define observable set of point of interest q_j as

$$O_j = \left\{ l_i \;\middle|\; abs(T_d^{\max}(q_j, l_i)) < e_j \;,\; abs(T_d^{\min}(q_j, l_i)) < e_j \right\}. \tag{9}$$

The inputs to the sensor placement technique are sets *Q, E* and *L* mentioned above. In the first step, the observability set of each q_j is calculated using (9) and our analytical model. Then, we find the optimum number of sensors and their locations such that there is at least one sensor in the observable set of each point of interest. This guarantees that the accuracy requirements are satisfied since any sensor placed on a grid cell in the observable set of a point of interest can sense temperature with the required accuracy. This problem is equivalent to the minimum hitting set problem which has been proven to be NP-complete. If $x_a = 1$ when the sensor is placed at grid cell a, and $x_a = 0$ otherwise, then the sensor minimization problem can be formulated as an integer linear program (ILP) as follows:

$$\begin{aligned} &\text{minimize} \quad \textstyle\sum_{a \in L} x_a \\ &\text{subject to} \quad \textstyle\sum_{a \in Oj} x_a \geq 1 \quad j = 1, \ldots, n \\ &x_a \in \{0, 1\} \quad \forall a \in L \\ &S = \{l_i | x_{l_i} = 1\}. \end{aligned} \tag{10}$$

Minimizing Σx_a (for $a \in L$) minimizes the total number of sensors while constraint $\Sigma x_a \geq 1$ (for $a \in O_j$) guaranties that there will be at least one sensor in observable set of hotspot j. For each grid with $x_a = 1$, a sensor is placed in the corresponding grid cell. To solve the ILP problem, we use *lp_solve* [26] which is based on the revised simplex and branch-and-bound methods.

Our proposed sensor placement technique minimizes the cost of sensors while maintaining the desired accuracy. However, our technique may not always be able to place enough sensors in the right locations due to design constraints, and even those sensors that are placed may degrade or fail during the lifetime of the system. These issues call for techniques to compensate for the lack of sensing hardware. Next section presents our accurate indirect temperature sensing technique which is able to estimate the temperature at locations not directly covered by sensors.

V. Indirect Temperature Sensing

Inaccuracies related to temperature sensing cannot always be effectively addressed at design time. There are multiple sources of errors; sensors fail over time, their output is subject to noise, and even if the system did not have sensor failures and noise, the location of hotspots dynamically changes under different workloads. To address such runtime issues, we propose indirect temperature sensing based on Kalman filtering which can accurately estimate the temperature at various locations on the die using inaccurate readings obtained from a limited number of noisy sensors. Our technique consists of a set of off-line setup steps shown in Fig. 3(a), followed by runtime procedure shown in Fig. 3(b).

The setup phase [Fig. 3(a)] starts by creation of chip's equivalent thermal *RC* network using models described in

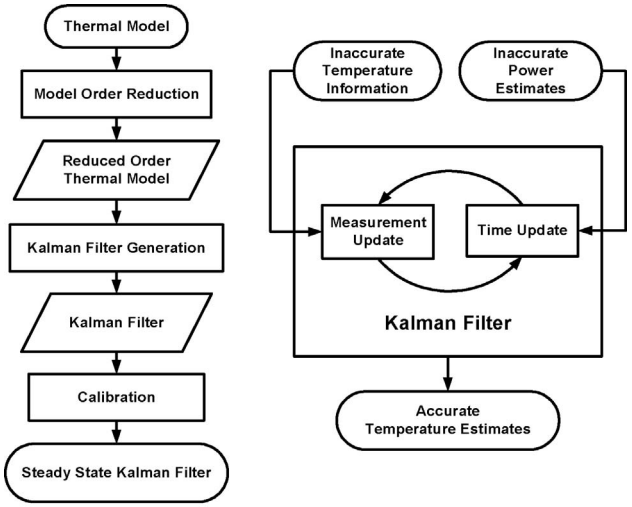


Fig. 3. Proposed technique. (a) Off-line setup. (b) Runtime temperature estimation by KF.

[8] and [9]. The linear dynamic system generated in this way is usually too large and too complex for an on-chip software implementation. Model order reduction is used to generate a much smaller yet accurate system. Calibration is performed by applying Kalman filter (KF) to the reduced order model of the system. The calibration ends when the KF reaches its steady state. The resulting steady-state KF is used during the normal operation to actually perform the temperature estimation [Fig. 3(b)].

The KF estimates the temperature in a predict-correct manner based on inaccurate information of temperature and power consumption. Time update equations project forward in time with the current temperatures and the error covariance estimates to obtain a priori estimates for the measurement step. The measurement update equations incorporate the new measurements into the a priori estimate to obtain an improved a posteriori estimate of the temperature. Details of the technique are explained in the next section.

A. Temperature Estimation

Temperature values at different locations on the die depend on various factors, such as power consumptions of functional units, layout of the chip and the package characteristics. Analysis and estimation of temperature requires a thermal model which represents the relation between these factors and the resulting temperature. The differential equations describing the heat flow have a form dual to that of electrical current. This duality is the basis for the micro-architectural thermal model proposed in [8] and is further explained in [1] and [9].

The lumped values of thermal R and Cs represent the heat flow among units and from each unit to the thermal package. We model the temperature at grid cell level [1] which enables more accurate and fine-grained temperature estimates. An analytical method is proposed in [1] to determine the proper size of a grid cell. The thermal network is represented in state space form with the grid cell temperatures as states and the power consumption as inputs to this system. The outputs of this state space model are the temperatures at the sensor locations which can be observed by sensor readings ($S(t)$). We define C_t and G_t as thermal capacitance and thermal conductance matrices, D as the input selection matrix which identifies the effect of power consumptions at current time steps on the temperature at next time step and F as the output matrix which identifies the sensor grid cells at which temperatures are observable. u is the vector of power consumption values for different components on the die and T is the vector of temperature values at different grid cells. The units for temperature and power are centigrade degree and watt. The system can be represented as

$$\begin{aligned} \frac{dT}{dt}(t) &= -C_t^{-1} G_t T(t) + C_t^{-1} D u(t) \\ S(t) &= F T(t). \end{aligned} \tag{11}$$

Since sensor measurements can be inaccurate and exact power consumption of each functional unit at runtime is not available, we use KF for accurate temperature estimation. The KF uses a form of feedback control to estimate a process in a predict-correct manner with time and measurement update phases. Time update equations project forward in time the current state of the system and the error covariance estimates to obtain a priori estimates for the measurement step. The measurement update equations incorporate the new measurements into the a priori estimate to obtain an improved a posteriori estimate.

We use Kalman filtering to both estimate the temperature and to filter out any thermal sensor noise. In order to apply the KF to our model, we convert the continuous time differential equations in (11) to corresponding discrete time equations in (12). Here, H, J, and F are the state matrix, input matrix and output matrix of the system, respectively. Furthermore, at time *n, T[n], u[n]*, and *S[n]* are the state vector representing temperatures at different grid cells, input vector of functional block power consumption and output vector of temperatures at sensor locations, respectively

$$\begin{aligned} T[n+1] &= H\,T[n] + J\,u[n] \\ S[n] &= F\,T[n]. \end{aligned} \tag{12}$$

Accurate estimation of power consumptions of each component at each time step is not practical in runtime. On the other hand, [11] shows that most of the energy in the power traces is concentrated in the DC component. The trend of temperature variations is determined by the average power over a period of time. This is especially true for power traces with very large DC components and smaller high-frequency harmonics [11]. Based on this fact, we use the average power consumption of each component as an estimation of the actual power consumption at that time.

Introduction of noise due to inaccuracies of modeling the process, *w[n]*, and the measurement noise, *v[n]*, enables us to rewrite the system formulation as

$$\begin{aligned} T[n+1] &= H\,T[n] + J\,u[n] + G w[n] \\ S_v[n] &= F\,T[n] + v[n]. \end{aligned} \tag{13}$$

The time-update equations for our system are given below. Here, $\check{T}[n|n-1]$ represents the estimate of $T[n]$ given the past measurements up to $S_v[n-1]$, $\check{T}[n|n]$ is the updated estimate based on the last measurement $S_v[n]$ and P is the error covariance matrix

$$\begin{aligned} \breve{T}[n+1|n] &= H\,\breve{T}[n|n] + J\,u[n] \\ P[n+1|n] &= HP[n|n]H^T + GQ[n]G^T. \end{aligned} \tag{14}$$

Given the current estimate $\breve{T}[n|n-1]$, the time update predicts the state value at the next sample $n+1$ (one step ahead). Then, the measurement update adjusts this prediction based on the new measurement $S_v[n+1]$. The measurement update equations for this system are

$$\begin{aligned} &\breve{T}[n|n] = \breve{T}[n|n-1] + M[n](S_v[n] - F\breve{T}[n|n-1]) \\ &M[n] = P[n|n-1]\,F^T(R[n] + FP[n|n-1]F^T)^{-1} \\ &P[n|n] = (I - M[n]F)\,P[n|n-1]. \end{aligned} \tag{15}$$

M is called Kalman gain or innovation gain. It is chosen to minimize the steady-state covariance of the estimation error given the noise covariance $Q = E(w[n]w[n]^T)$ and $R = E(v[n]v[n]^T)$. Computational complexity of the KF is $O(k^3)$ due to the matrix inversion in calculating Kalman gain $M[n]$, where k is the size of the dynamic model. The next section describes the methods we exploit in order to speed up the runtime computation.

B. Reducing Computational Complexity

We introduce two techniques that significantly reduce the computational complexity of our model. One of the techniques reduces the size of the model used in KF, while the other reduces the number of computations required for KF.

1) *Steady-State Kalman Filtering:* The time scale at which the sensor noise characteristics change is much larger than the time scale at which we study the system (months or years compared to seconds). Thus, we assume the system and noise covariances are time-invariant. As a result, we can use steady-state KF in which it is not necessary to compute the estimation error covariance or Kalman gain in real time [12]. The steady-state KF reduces the computational overhead from $O(k^3)$ to $O(k^2)$ while still providing good accuracy [12]. A calibration step is needed prior to runtime operation in order to get the KF to steady state. Running the KF during the calibration step makes gain and covariance matrices converge to constant values. In our experiments, we show that use of steady-state KF reduces the computational complexity to several orders of magnitude without significant effect on accuracy.

2) *Model Order Reduction:* The model order reduction enables us to find a low-dimensional but accurate approximation of the thermal network which preserves the input-output behavior to a desired extent. We use a projection based implicit moment matching method (PRIMA) [14] which is used to find a mapping from the high-dimensional space of the given state-space model to a lower dimensional space. In this technique, Krylov subspace vectors are used instead of moments. For a square matrix A of dimension N and a vector b, the subspace spanned by the vectors $[b, Ab, \ldots, A^{q-1}b]$ is called a Krylov subspace of dimension m generated by $\{A, b\}$ and is denoted by $Kr(A, b, q)$.

With thermal capacitance and conductance matrices represented by C_t and G_t, respectively, the circuit formulation shown in (11) can be represented in the form

$$sC_tT = -G_tT + Du. \tag{16}$$

The reduced order model is generated using congruence transformation, where $C_r = V_q^T C_t V_q$, $G_r = V_q^T G_t V_q$, $D_r = V_q^T D$, $T_r = V_q^T T$:

$$sC_rT_r = -G_rT_r + D_ru. \tag{17}$$

The projection matrix $V_q = \{V_1, V_2, \ldots, V_q\}$ is obtained by Arnoldi process such that

$$\begin{aligned} &Span\{V_1, V_2, \ldots, V_q\} = Kr\{-G_t^{-1}C_t, -G_t^{-1}D, q\} = \\ &Span\{-G_t^{-1}D, (-G_t^{-1}C_t) \times -G_t^{-1}D, \ldots, (-G_t^{-1}C_t)^{q-1} \\ &\qquad \times -G_t^{-1}D\} \end{aligned} \tag{18}$$

and

$$\begin{aligned} V_i^T V_j &= 0 \quad \text{for all } i \neq j \\ V_i^T V_i &= 1 \quad \text{for all } i. \end{aligned} \tag{19}$$

This approach matches moments up to order q. The larger the number of matched moments, the closer is the behavior of the reduced order model to the original system, but at the cost of higher processing time. Because of the moment-matching properties of Krylov-subspaces, the reduced transfer function will agree with the original up to the first q derivatives on an expansion around some chosen point in the complex plane (usually $s = 0$). In addition, due to the congruence transformation, the reduced model inherits the structure of the original model, which means the passivity is preserved. Interested readers can refer to [14] for a detailed discussion of the technique.

There are other model order reduction techniques which are designed for linear circuits with multiple sources. For example, energy kernel system (EKS) [15] can match higher moments compared to PRIMA in multiple-input multiple-output systems, but imposes limitations on the inputs and, more importantly, incurs higher computational overhead. Our experimental results show that PRIMA works well for our applications since our network consists of only simple linear elements (*Rs* and *Cs*). Moreover, the topology of the network is such that it operates as a low pass filter which eliminates the high frequencies of the inputs. Therefore, PRIMA with a few moments around frequency $s = 0$ provides sufficient accuracy and acceptable overhead compared to RHS-model order reduction methods like EKS [15]. The effectiveness of PRIMA is shown in Table V.

C. Detecting Sensor Failure and Degradation

As explained in the previous sub-sections, one of the techniques we use for real time realization of the technique is the use of steady-state KFs. Steady-state KF is optimal when assuming stationary noise. The fact that the noise characteristics of the thermal sensors do not change rapidly, makes steady-state KF applicable to our problem. Sensor degradations or failures which cause these changes usually happen at the order of months. While steady-state Kalman filtering works well for the stationary noise characteristics, it is not guarantied to work well under non-stationary noise. Therefore, the changes in the characteristics of the sensor noise affect the accuracy of indirect temperature sensing. Here, we address this problem by proposing a technique to detect the variations in the

characteristics of sensor noise in order to detect the sensor degradation or failure before they affect the accuracy of the technique. Then, these variations are addressed by adapting the indirect temperature sensing to these variations.

The steady-state KF is completely dependent on the characteristics of the noise. Therefore, a KF generated for a certain set of noise characteristics might not work well for a different set. When a change is detected, the calibration phase of the technique is performed again and a new steady-state Kalman filter is generated based on the current noise characteristics. In order to detect these variations, we use a hypothesis test called sequential probability ratio test (SPRT) which is specifically designed for sequentially collected data. It allows us to detect the variations in the statistical characteristics of the estimation error. An alternative would be setting thresholds on mean of the errors, but this can detect the variations only after they have happened. SPRT is able to detect the abrupt changes as threshold-based techniques, but it is also able to detect slow gradual variations which evolve over a long period of time [29]. It has been shown in [30] that a hypothesis test based on the SPRT is optimal in the sense that for given false and missed alarm probabilities α and β, it results in the lowest possible false or missed alarms. As stated before, α and β are the probabilities of false positives and false negatives, respectively.

SPRT is based on a pair of hypotheses, H_0 and H_1 which are null hypothesis and alternative hypothesis, respectively. The null hypothesis (H_0) states that the estimation errors are drawn from a distribution with mean zero and standard deviation of σ. The alternate hypothesis states that the estimation errors are drawn from a distribution with mean μ and standard deviation of σ. In other words, H_0 assumes no change in the characteristics of the noise, while H_1 implies variations in these characteristics. The decision between these two hypotheses is based on the cumulative sum of the log-likelihood ratio (Λ_i)

$$S = S_i + \log \Lambda_i. \tag{20}$$

H_1 is accepted if $S_i \geq b$, H_0 is accepted if $S_i \leq a$, otherwise the monitoring continues, where a and b are chosen as two sub-equations:

$$a = \log\left(\frac{\beta}{1-\alpha}\right) \text{ and } b = \log\left(\frac{1-\beta}{\alpha}\right) \tag{21}$$

where α is the false alarm probability, which is the probability of accepting H_1 when H_0 is true, and β is the missed alarm probability, which is the probability of accepting H_0 when H_1 is true. α and β are decided in advance by the user.

Likelihood ratio is the ratio of the maximum probability of a result under two different hypotheses. In other words, likelihood ratio can be calculated as the maximum probability of H_0 (the estimation errors are drawn from a distribution with mean zero and standard deviation of σ) to H_1 (estimation errors are drawn from a distribution with mean μ and standard deviation of σ) given the current observations

$$\Lambda_n = \frac{\Pr(\varepsilon_1, \ldots, \varepsilon_n | H_1)}{\Pr(\varepsilon_1, \ldots, \varepsilon_n | H_0)} \tag{22}$$

when n is the number of observations and $Pr(\varepsilon_1, \ldots, \varepsilon_n | H_0)$ and $Pr(\varepsilon_1, \ldots, \varepsilon_n | H_1)$ are the probability of observing sequence $\varepsilon_1, \ldots, \varepsilon_n$ under H_0 and H_1, respectively. Assuming independent observations, we can write

$$\Lambda_n = \frac{\prod_{i=1}^{n} \Pr(\varepsilon_i | H_1)}{\prod_{i=1}^{n} \Pr(\varepsilon_i | H_0)} = \prod_{i=1}^{n} \frac{\Pr(\varepsilon_i | H_1)}{\Pr(\varepsilon_i | H_0)} \tag{23}$$

and

$$\log \Lambda_n = \sum_{i=1}^{n} \log \frac{\Pr(\varepsilon_i | H_1)}{\Pr(\varepsilon_i | H_0)}. \tag{24}$$

When operating at runtime mode, the technique monitors the temperature estimation error ε (difference between the estimated and observed temperature). With every new observation, (22) is calculated and the new log likelihood ratio is compared against the thresholds a and b. If the new log likelihood ratio exceeds b, SPRT reports degradation, while if the likelihood ratio gets under a, SPRT assumes the sensor is working fine. The changes are detected before they are large enough to affect the indirect temperature sensing results. Due to extremely low rate of sampling required in this technique (due to the extremely slow changes in the noise characteristics), we are not concerned about the overhead of the technique, but for estimation errors which are normally distributed, [29] proposes a compact expression with very low computational overhead. As stated previously, the probabilities α and β are defined by the user. When a decision is made about the H_0 or H_1, the technique starts at $n = 1$ with current observation. For our experiments, we used $\alpha = 0.01$, $\beta = 0.0001$, and sampling interval of 1 min.

In the next section, we discuss how our techniques complement each other thorough various experiments.

VI. Experimental Results

For our experiments, we use two different multi-processor SoCs: 1) SoC1 consisting of 6 *XScale* cores implemented in 90 nm process [16] whose layout is shown in Fig. 1, and 2) SoC2 consisting of two SPARC cores and 4 *XScale* cores implemented in 65 nm process whose layout is shown in Fig. 4 where the cores share the L2 cache. A set of programs from the automotive/industrial, network and telecommunications categories of MiBench [17] are selected and run on datasets provided by [18]. Moreover, we have also used a set of programs from Mediabench [40] benchmark suite and run them on SPARC cores. Pareto distribution is used [19] to introduce idle intervals between the MiBench tasks. A timeout-based dynamic power management policy is applied in order to determine the active and low power states which each core experiences during running these workloads. Power values for *XScale* are measured on a real system and scaled for the target process. Power values for Alpha cores are calculated using Wattch power estimator [27] integrated with M5 simulator [28]. HotSpot 3.0 [13] grid mode is used for thermal simulations. Each processor is assumed to have its own L2 cache. Parameters used for package are: convection capacitance 140.4 J/K, convection resistance 0.1 K/W, spreader thickness 10^{-3} m, and initial temperature of 333 °K.

First, we present the experimental results on the offline parts of our techniques which are performed during the design of the

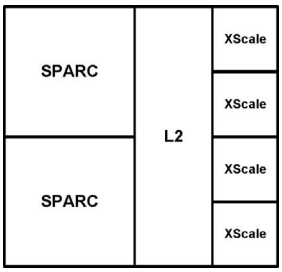


Fig. 4. Layout of SoC2.

system. These include our analytical model for upper bound on on-chip temperature differences and our sensor placement algorithm. Then, the results of the runtime techniques are discussed which are performed during the normal operation of the system. These include indirect temperature sensing and also detection of sensor failure and degradation.

A. Offline Techniques

1) *Maximum Temperature Difference Model:* Our proposed model analytically calculates the upper bound on temperature differences on the die eliminating the high overhead of extensive simulations. Very specific combinations of conditions may be required to achieve the maximum temperature difference on the die. Such specific conditions may not be created by the benchmarks, but may happen during the actual operation of the system. Our model can generate a set of input power traces which create such a maximum temperature difference scenario.

Figs. 5–7 provide an example of this for SoC1. Fig. 5 shows a simulation slice in which the temperature difference between points a and b has reached its highest value. The dotted line shows the maximum temperature difference estimated by our model which is clearly higher than the benchmark results. Fig. 6 shows the situation in which the actual temperature difference found by our method exceeds the maximum reported by simulations using standard benchmarks. The first row shows $h_{(a,b),i}(t)$ which is the response of the temperature difference between a and b to the impulse applied at input i. $h_{(a,b),i}(t)$ is calculated by differentiating the response of the temperature difference to the step input i (since it is in discrete time, this is done by differencing the consecutive samples of step response). To keep the example easy to follow, we considered only three power sources of SoC1 and the rest are turned off. The power traces generating the maximum temperature differences are calculated as explained in Section III. Applying each of these traces leads to the corresponding temperature differences shown in the third row. The overall maximum temperature difference due to all power sources occurs at time 0.9 s (time unit 900) as shown in Fig. 7.

As shown in this example, the maximum temperature differences can be much larger than what is observed under standard benchmarks. Therefore, when we need to know the maximum temperature differences on the die, running benchmarks may not be enough and models such as ours are needed to get accurate data. The difference between simulations and the model in this particular example is not large because of the low power consumption of *XScale* cores and the fact that for simplicity we looked at only 3 cores with observation points in proximity

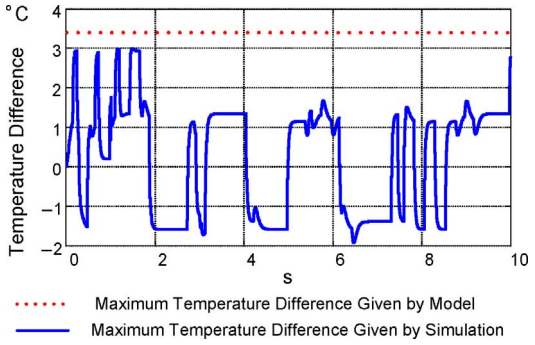


Fig. 5. Temperature difference between points a and b.

of each other. The error can be much larger in high-end processors with higher powered devices. Table I shows that errors as high as 9 °C occur in estimates of temperature differences when relying only on simulations. Even using combinations of different benchmarks does not resolve the problem. Such errors can cause significant functional and reliability issues. For example, if the maximum temperature difference between a sensor and a hotspot is underestimated, it may cause late activation of DTM which can result in serious reliability problems.

The simulation overhead of our method is insignificant. As explained in Section III, this method involves simulating one step input for each power source in the worst case compared to simulating the whole set of benchmarks when standard simulation is used. If linear systems analytical methods are used to calculate the step response, then there will be no simulation overhead.

2) *Sensor Placement:* The sensor placement method we developed uses our analytical model for maximum on-die temperature differences. The experimental results show that it reduces the number of required sensors as compared to previous work. Previous techniques such as [21] and [22] depend on calculation of the range of the hotspot which is the maximum distance r from the hotspot that a sensor can be placed while still maintaining the intended accuracy.

This range has a circular form and is centered at the point of interest which limits the accuracy and the efficiency of such techniques. Some points which meet the accuracy requirements might be missed, as demonstrated on SoC with 6 *XScale* cores shown in Fig. 8. The two *X*'s represent the points of interest to be monitored. The observability areas of the points of interest are shown by solid lines. Circular ranges of the hotspots are shown by dotted circles. Our sensor placement technique considers the observability area of a point of interest instead of its circular range as the potential location of a sensor. Therefore, it can identify the point marked by * at the overlapping part of the observability areas to place a single sensor to monitor both points of interest. When using the circular ranges, this sensor location would not be identified since the ranges do not overlap; and therefore, two separate sensors would be required. We evaluate our sensor placement algorithm for different values for the desired accuracies and compared it with techniques that uses circular ranges [21]. In [21], only the sensor placement model is proposed and actual results on sensor placement are not provided. Moreover, the activity factor parameter (f_a) which is one of the important

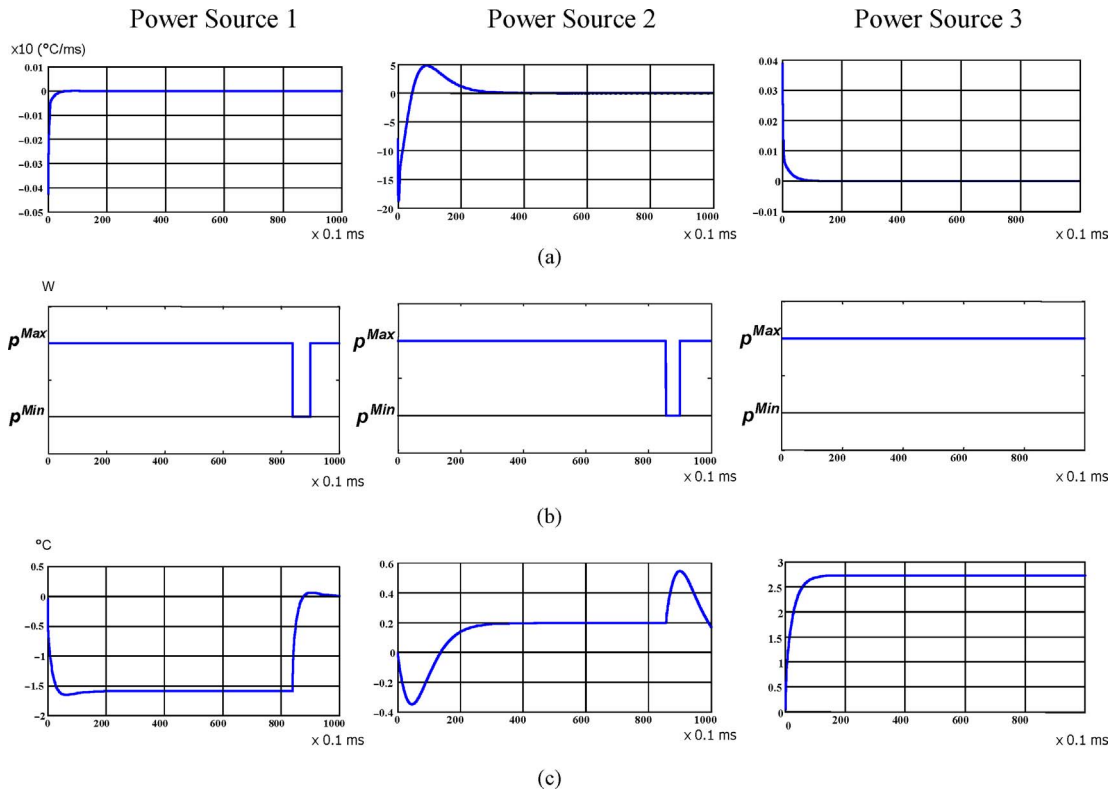


Fig. 6. Generating the maximum temperature difference by constructing the proper power trace. (a) $h_{(a,b),i}(t)$. (b) Power traces leading to maximum temperature difference. (c) Temperature difference due to generated power trace.

parameters in the proposed model in that work depends on the workload and the paper does not explain in enough details that how this parameter is calculated. Due to these reasons, we were not able to reproduce the actual sensor placement results of this paper for comparison. Therefore, in order to be fair, we assumed the best results which this model could achieve. We found the largest possible circular range this model could have found. Then, we used the outcomes of this model in our sensor placement algorithm. In other words, we are comparing our techniques to the best possible results the other model could achieve under the best conditions.

Table II shows the number of sensors required to monitor eight locations of interest on SoC1 and eight locations on SoC2 for specified maximum tolerable error between the temperature sensor and the actual temperature at the point of interest (accuracy in Table II). As these tables show, our sensor placement technique is able to reduce the number of sensors needed more aggressively with lower desired accuracy than previous work [21]. It should be mentioned that increasing the number and locations containing the points of interest will improve the efficiency of our technique. Our sensor placement technique minimizes the number of sensors by sharing a sensor among multiple points of interest.

This technique takes advantage of the overlapping observability regions of the locations of interest in order to find the possible sharing of a sensor among various locations of interest. Increasing the number of the locations of interest increases the chances that observability regions of these locations overlap, so more sensors could be shared among various points of interests. This would help reduce the number of sensors.

TABLE I
ERRORS IN TEMPERATURE DIFFERENCE SIMULATIONS (°C)

	Positive			Negative		
Benchmarks	Mean	Std. Dev.	Max.	Mean	Std. Dev.	Max.
MiBench (Automotive)	1.09	0.78	4.34	1.01	0.72	3.77
MiBench (Network)	6.89	2.60	9.39	6.96	0.55	9.55
MiBench (Telecomm.)	6.02	2.53	9.16	6.36	2.47	9.29
MiBench (Mixed)	1.15	1.15	8.05	0.59	0.55	7.08
MediaBench (Mixed)	3.13	2.68	8.36	2.75	2.27	7.41

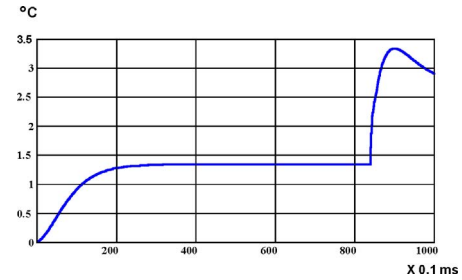


Fig. 7. Temperature difference (°C) for Fig. 6.

B. Online Techniques

1) *Indirect Temperature Sensing:* Unlike design time techniques discussed above, our method for accurate indirect temperature sensing addresses runtime issues such as limited number of available sensors, their degradation or failure, dynamic changes in location of hotspots, etc. Indirect temperature sensing requires an offline setup step which has been implemented in MATLAB, while the runtime part of the algorithm is implemented in C++ on XScale or SPARC processors.

Temperature values are obtained by running HotSpot [13] in grid mode with XScale power measurements as inputs.

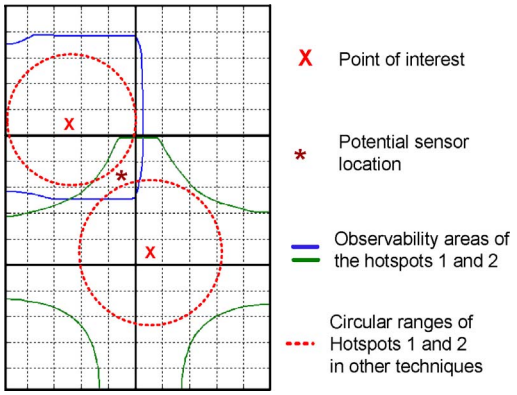


Fig. 8. Using observability area vs. circular range.

The temperature values of the grid cell containing the sensors are observable, while the temperature at other grid cells are assumed to not be observable and must be estimated using our technique. For our experiments, we used a 18×12 grid for SoC1 and a 20×20 grid for SoC2.

One of the advantages of using the PRIMA model order reduction technique for indirect sensing is that the size of reduced model depends only on the number of power sources and the number of matched moments, not the number of grid cells. Therefore, increasing the granularity of the grid in order to increase the accuracy does not result in higher computational overhead. An analytical method presented in [1] is used to find the appropriate number and size of the grid cells.

No specific sensor technology is assumed in this paper. The readings from the temperature sensors are used as starting temperature values for our model. Gaussian noise has been superimposed on the actual temperature values to model the inaccuracies of real thermal sensors. Processes generating noise are assumed to be stationary between off-line calibrations.

Fig. 9 shows that the estimated temperature values closely follow the actual temperature at the location of interest on SoC1. Accurate estimates of the temperature are important to prevent early or late activation of DTM techniques due to sensor noise and errors. Indirect temperature sensing is very accurate in estimating temperature at locations far away from a limited number of sensors available on the die as shown in Table III. It should be noted that the quality of estimation using KF depends on many factors such as workload characteristics, number and location of the sensors and the characteristics of noise. For example, having more sensors located closer to the locations of interest would result in better estimation. For our application, as our experiments in Table III show, even with a very few sensors, our technique is able to estimate the temperature with reasonable accuracy.

For SoC1, we estimate the temperature at six locations of interest using only 2, 3, 4, and 5 sensors. For SoC2, we estimate the temperature at eight locations of interest using 2, 3, 5, and 7 sensors. Each sensor is equidistant to the hotspots it must cover. In order to reduce the inaccuracies due to step size and model reduction, the step size is chosen to be 10 ms and three moments are matched. The mean absolute error and its standard deviation are reduced by up to an order of magnitude.

Another important parameter affecting both accuracy and computational requirements of our technique is the time step at which temperature sensors are read and KF is applied. Table III shows statistics of measurement and estimation errors for different sizes of time steps used on SoC1 and SoC2. The basic time step is chosen at 10^{-4} s and multiplied by powers of 2. For this experiment, in order to reduce the inaccuracies due to sensor model order reduction, three moments are matched. One sensor monitors each location of interest, but this sensor is placed at an arbitrary location around the hotspot to show that the technique does not depend on the relative position of the sensor and the location of interest. Step size between 10 ms and 100 ms provide reasonable accuracy.

Table IV also shows the effect of the indirect temperature sensing technique on the performance impact of DTM techniques. It compares the relative slowdown of a DTM technique when it is driven by temperature values read directly from the sensors and temperature estimates of our indirect temperature sensing. The DTM technique used here is a threshold based DTM technique which reduces the voltage/frequency of a core when its temperature exceeds the threshold of 75 °C.

Slowdown is compared by looking at the number of instructions executed per second at high utilizations. As this table shows, our indirect temperature sensing technique can reduce the slowdown due to DTM by more than 6X. This is due to the fact that more accurate temperature estimates reduce the number of false positives in triggering the DTM technique which reduces the slowdown due to DTM.

This figure also shows that since larger step sizes reduce the accuracy of the indirect temperature sensing, the reduction in DTM slowdown caused by indirect sensing reduces as step sizes get larger.

Table V shows how matching different number of moments affects the accuracy of our technique. The step size is set at 50 ms. Size of the reduced model is the number of functional units times the number of matched moments. For SoC1 example, since there are six power consuming functional units, matching 4, 3, 2, and 1 moments reduces the model size to 24, 18, 12, and 6, respectively.

As it is shown in the table, after matching the second moment, further moment matching does not significantly improve accuracy. Matching two or three moments provides sufficient accuracy for most applications. As explained in the previous sections, an important step toward online realization of indirect temperature sensing is using steady-state Kalman filtering.

We next analyze the overhead of our indirect temperature estimation technique. During the calibration process, i.e., before the KF reaches its steady state, we use general KF. Since calibration can be performed offline, the overhead of the general KF does not affect performance. The number of calibration steps, in our case less than 100, depends on the thermal network and noise characteristics. The runtime overhead is shown in Fig. 10. The cost of using regular KF is compared with the steady-state KF used by online portion of our indirect temperature sensing technique. As can be seen in Fig. 10, the performance overhead of steady-state KF is significantly lower. This difference grows for larger models. Using model order reduction and steady-state KF allow performing indirect temperature sensing even on *XScale* on which floating point instructions have to be emulated due

TABLE II

NUMBER OF SENSORS NEEDED BY OUR TECHNIQUE AND RANGE-BASED METHODS

	SoC1							SoC2						
Accuracy (°C)	1	2	3	4	5	6	7	1	2	3	4	5	6	
Our technique	7	7	6	5	5	5	4	8	8	7	6	6	5	5
Circular range [21]	7	7	7	7	6	6	6	8	8	8	7	7	6	6

TABLE III

ERROR STATISTICS FOR LIMITED NUMBER OF SENSORS

	SoC1				SoC2				
	Temp. Measurement Errors (°C)		Temp. Estimation Errors (°C)			Temp. Measurement Errors (°C)		Temp. Estimation Errors (°C)	
Sensor No.	Mean Abs. Error	Std. Dev.	Mean Abs. Error	Std. Dev.	Sensor No.	Mean Abs. Error	Std. Dev.	Mean Abs. Error	Std. Dev.
2	3.74	4.72	0.77	1.28	2	5.61	6.02	1.26	2.35
3	3.72	4.60	0.76	1.27	3	4.17	6.41	1.35	2.26
4	4.41	5.50	0.75	1.27	5	4.34	5.76	1.12	1. 50
5	3.29	3.94	0.76	1.27	7	5.03	6.60	0.85	1.23

TABLE IV

ERROR STATISTICS FOR DIFFERENT TIME STEPS

	SoC1						SoC2					
Step Size (10^{-4}s)	Sensor Measurement Errors (°C)			Indirect Temperature Sensing Errors (°C)			Sensor Measurement Errors (°C)			Indirect Temperature Sensing Errors (°C)		
	Mean Absolute Error	Std. Dev.	DTM Slowdown	Mean Absolute Error	Std. Dev.	DTM Slowdown	Mean Absolute Error	Std. Dev.	DTM Slowdown	Mean Absolute Error	Std. Dev.	DTM Slowdown
128	2.91	4.04	40%	0.13	0.25	6%	3.94	4.75	55%	0.54	0.62	9%
512	3.09	4.33	41%	0.49	0.89	24%	3.66	4.82	52%	1. 12	1.64	22%
1028	4.29	4.27	44%	0.97	1.24	28%	4.04	4.38	49%	1.77	2.15	34%
2056	3.89	5.08	43%	1.74	1.39	38%	3.87	4.51	51%	2.66	3.34	46%

TABLE V

EFFECT OF NUMBER OF MATCHED MOMENTS ON TEMPERATURE ESTIMATION ERROR

Sensor	No. of Matched Moments	SoC1			SoC2		
		Model Size	Mean Abs. Error	Std. Dev.	Model Size	Mean Abs. Error	Std. Dev.
Measure. Error (°C)	–	–	3.38	4.82	–	5.64	6.60
Estimation Error (°C)	1	6	0.88	1.24	7	1.78	2.28
	2	12	0.75	1.05	14	1.53	1.79
	3	18	0.68	0.90	21	1.24	1.67
	4	24	0.48	0.88	28	0.69	0.92

to the lack of floating point units. The runtime overhead on *XScale* can be further reduced by using a fixed point implementation, or running the technique on a processor that has a floating point unit. The reduction in overhead due to use of floating point unit is clearly shown in Fig. 10(b) where the execution times are reported on a SPARC processor. As the figure shows, for the MPSoCs used in our experiments, the runtime overheads are in the order of 100 μs.

2) *Detecting Sensor Failure and Degradation:* Here, we show how using a steady-state KF while the noise characteristics are not stationary can affect the accuracy of the results. To prevent such inaccuracies, we use a sequential hypothesis test called SPRT described in the previous sections. This allows us to detect the meaningful changes in the sensor noise characteristics before they affect the accuracy of the technique. Fig. 11 shows how SPRT detects the meaningful changes in the estimation errors due to the changes in the characteristics of the noise and triggers adaptation of the technique to address the new changes.

Thermal sensors are usually based on temperature sensitive diodes, and according to [31], the most common faults in diodes are open, short and degradation. Open and short failures in the diode cause constant sensor readings which according to test community, we call it stuck at fault. We tried stuck at faults with various values around the average

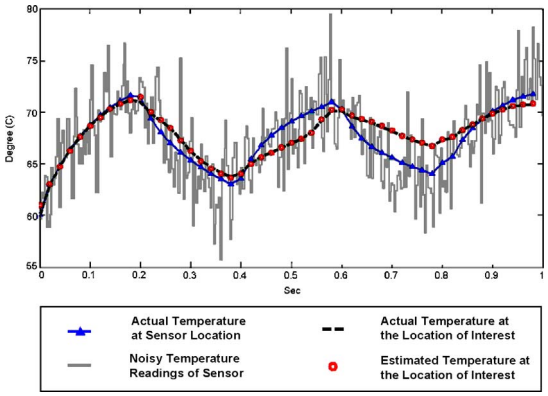


Fig. 9. Comparison of sensor, actual, and estimated temperatures.

expected sensor reading. Using this technique, all of these failures were detected in <1 h.

Another common fault in these diodes is degradation which means the deviation of parameters from the expected values. In order to emulate subtle incipient sensor degradations, we apply a gradual change in the average of the error which changes this average by a couple of degrees in a year. α and β values are set to 0.01 and 0.0001, respectively. This will result in a threshold of 100 for the likelihood ratio. The y-axis shows the likelihood ratio (Λ_i) and the threshold for deciding about the degradation.

TABLE VI
EFFECTS OF SENSOR DEGRADATION AND FAILURE

Sensor Degradation Type	SoC1				SoC2			
	Estimation Error		DTM Slowdown	Detection Time By SPRT	Estimation Error		DTM Slowdown	Detection Time By SPRT
	Mean Absolute Error	Std. Dev			Mean Absolute Error	Std. Dev		
Gradual mean change (2 °C/year)	0.59	0.61	19%	~15 weeks	1. 35	1.55	25%	~17 weeks
Gradual mean change (1 °C/year)	0.53	0.67	16%	~ 8 weeks	1.33	1.45	24%	~ 10 weeks
Bias (1 °C)	1.15	1.60	40%	~ 1.5 h	1.86	2.54	36%	~ 2 h
Bias (0.5 °C)	0.85	1.10	26%	~ 2 h	1.79	2.35	35%	~ 3 h
Stuck to constant value	1.47	2.63	54%	<1 h	2.89	3.47	48%	<1 h

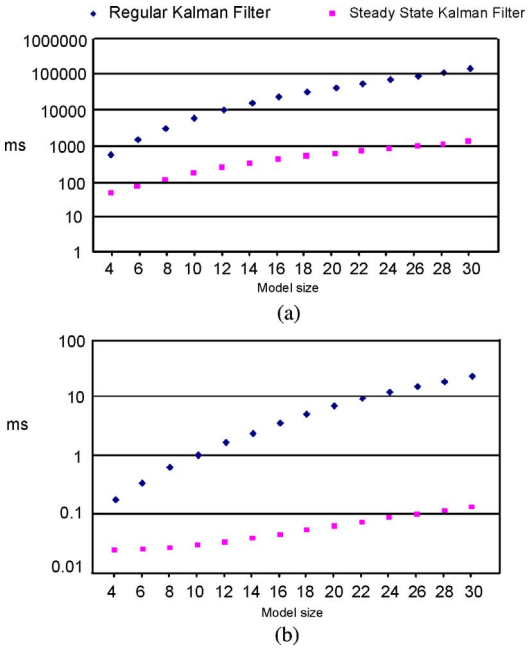


Fig. 10. Runtime of the technique on (a) *XScale* and (b) SPARC.

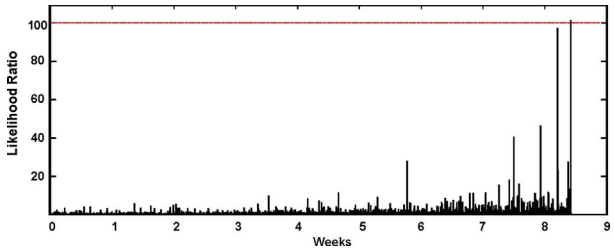


Fig. 11. SPRT technique to detect sensor degradation.

As shown in the figure, the likelihood ratio has passed the threshold at about two months. This triggers an alarm showing degradation of the sensor and the indirect sensing is signaled to adapt itself to this change and address this degradation by regenerating the steady-state KF. We assume different cases of sensor degradation or failure by adding some error to the sensed temperature. These error types include an error signal whose mean changes gradually and very slowly from 0 to 1 or 2 in a year. Another error introduced is a constant bias (0.5 °C and 1 °C) in the sensor readings. Another type of failure considered is stuck at error which means reading the same value from the sensor irrespective of the actual temperature.

Table VI shows how such sensor degradations will affect the estimation error and DTM slowdown before these sensor degradations are detected and addressed by regenerating the steady-state KF. It also shows the time it takes to detect these degradations.

As in the example shown in Fig. 11, the sampling time of the sensor degradation technique is 1 min and α and β values are set to 0.01 and 0.01. As the table shows, for the gradual changes in the error characteristics, the error is detected early before it affects the accuracy of the technique. For example, for a gradual mean change of 1 °C/year, the error is detected after about two months before it can cause significant DTM slowdown. For more significant and abrupt changes, such as bias values or stuck at errors which cause significant effect on DTM, the technique is able to detect the error in a matter of hours or so. This quick detection of more severe degradations minimizes the effect of these inaccuracies on the DTM slowdown. Another important point to note is that these detection times can be reduced with more frequent sampling. The cost will be more frequent computations for SPRT which usually are not significant since the technique can be implemented by accessing lookup tables. Using (24) in general cases implies two lookups in the look up table, one division, one logarithm operation and one addition. For the normally distributed observations, [29] suggests a compact expression which can be implemented by just two additions for each sample.

VII. CONCLUSION

We proposed two techniques for obtaining accurate temperature information from thermal sensors. The first one is a design time technique for thermal sensor allocation and placement which is able to guarantee a maximum sensor placement error. This technique is based on our model for finding the maximum temperature difference between arbitrary points on the chip. The model can be utilized in other application such as estimating the maximum temperature variations on the chip. Compared to previous work, our sensor placement technique can reduce the number of sensors needed by 16% on average while guaranteeing the specified sensor accuracy. The second technique, called indirect sensing, is a runtime technique for accurate estimation of temperature at different locations of the chip based on inaccurate values obtained from a limited number of noisy on-chip temperature sensors. We also proposed a technique to detect the sensor degradation and failure which we use to trigger the calibration of indirect temperature sensing and addressing these inaccuracies in the technique. Our experimental results showed an order of magnitude reduction in the maximum temperature estimation error using indirect temperature sensing.

REFERENCES

[1] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling method for CMOS VLSI systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[2] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in VLSI circuits: Principles and Methods," in *Proc. IEEE, Spec. Issue Thermal Anal. ULSI*, vol. 94, no. 8, pp. 1487–1501, Aug. 2006.

[3] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. Int. Symp. High-Perform. Comput. Architect.*, 2001, pp. 171–182.

[4] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. DATE*, Apr. 2007, pp. 1659–1664.

[5] A. Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and thermal management in the Intel™ Core™ Duo Processor," *Intel Technol. J.*, vol. 10, no. 2, pp. 109–122, May 2006.

[6] E. Rotem, J. Hermerding, C. Aviad, and C. Harel, "Temperature measurement in the Intel® Core™ duo processor," in *Proc. Int. Workshop Thermal Investigat. ICs*, Sep. 2006, pp. 23–27.

[7] K. Whisnant, K. C. Gross, and N. Lingurovska, "Proactive fault monitoring in enterprize servers," in *Proc. IEEE Int. Multiconf. Comput. Sci. Comput. Eng.*, Jun. 2005, pp. 3–10.

[8] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature aware microarchitecture," in *Proc. Int. Symp. Comput. Architect.*, Jun. 2003, pp. 2–13.

[9] W. Huang, M. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy, "Compact thermal modeling for temperature-aware design," in *Proc. Design Automat. Conf.*, Jun. 2004, pp. 878–883.

[10] D. Fetis and P. Michaud, "An evaluation of hotspot-3.0 block-based temperature model," in *Proc. WDDD*, Jun. 2006.

[11] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, "Fast thermal simulation for runtime temperature tracking and management," *IEEE Trans. Comput. Aided Design Integr. Circuits*, vol. 25, no. 12, pp. 130–136, Dec. 2006.

[12] D. Simon, *Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches*. New York: Wiley, 2006.

[13] *HotSpot* [Online]. Available: http://lava.cs.virginia.edu/HotSpot/

[14] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: Passive reduced order interconnect macro-modeling algorithm," *IEEE Trans. Comput. Aided Design*, vol. 17, no. 8, pp. 645–654, Aug. 1998.

[15] J. M. Wang, and T. Nuyen, "Extended Krylov subspace method for reduced order analysis of linear circuits with multiple sources," in *Proc. Design Automat. Conf.*, Jun. 2000, pp. 247–252.

[16] *Intel PXA270 Processor, Electrical, Mechanical and Thermal Specification Data Sheet* [Online]. Available: http://www.intel.com

[17] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Ann. Workshop Workload Character.*, Dec. 2001, pp. 3–14.

[18] G. Fursin, J. Cavazos, M. O'Boyle, and O. Temam, "MiDataSets: Creating the conditions for a more realistic evaluation of iterative optimization," in *Proc. Int. Conf. HiPEAC*, Jan. 2007, pp. 245–260.

[19] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Event-driven power management," *IEEE Trans. Comput.-Aided Design*, vol. 20, no. 7, pp. 840–857, Jul. 2001.

[20] R. Mukherjee and S. O. Memik, "Systematic temperature sensor allocation and placement for microprocessors," in *Proc. Design Automat. Conf.*, 2006, pp. 542–547.

[21] K.-J. Lee and K. Skadron, "Analytical model for sensor placement on microprocessors," in *Proc. IEEE ICCD*, Oct. 2005, pp. 24–27.

[22] S. Mondal, R. Mukherjee, and S. O. Memik, "Fine-grain thermal profiling and sensor insertion for FPGAs," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, p. 4390.

[23] A. Coskun, T. S. Rosing, K. Mihic, G. De Micheli, and Y. Leblebici, "Analysis and optimization of MPSoC reliability," *J. Low Power Electron.*, vol. 2, no. 1, pp. 56–69, Apr. 2006.

[24] C. J. Lasance, "Thermally driven reliability issues in microelectronic systems: Status quo and challenges," *Microelectron. Reliabil.*, vol. 43, no. 12, pp. 1969–1974, 2003.

[25] M. Cho, S. Ahmedtt, and D. Z. Pan, "TACO: Temperature aware clock-tree optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2005, pp. 582–587.

[26] *LP-Solve* [Online]. Available: http://lpsolve.sourceforge.net/

[27] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. Int. Symp. Comput. Archit.*, 2000, pp. 83–94.

[28] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The M5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul.–Aug. 2006.

[29] K. Gross, K. Whisnant, and A. Urmanov, "Electronic prognostics through continuous system telemetry," in *Proc. 60th Meeting Soc. MFPT*, Apr. 2006, pp. 53–62.

[30] K. Fu, *Sequential Methods in Pattern Recognition and Machine Learning*. New York: Academic Press, 1968.

[31] P. Kabisatpathy, A. Barua, and S. Sinha, *Fault Diagnosis of Analog Integrated Circuits*. Berlin, Germany: Springer, 2005.

[32] S. Sharifi and T. S. Rosing, "An analytical model for the upper bound on temperature differences on a chip," in *Proc. Great Lakes Symp. VLSI*, 2008, pp. 417–422.

[33] S. Sharifi, C. Liu, and T. S. Rosing, "Accurate temperature estimation for efficient thermal management," in *Proc. Int. Symp. Quality Electron. Design*, 2008, pp. 137–142.

[34] J. Long, S. O. Memik, and G. Memik, "Thermal monitoring mechanisms for chip multiprocessors," *ACM Trans. Architect. Code Optim.*, vol. 5, no. 2, pp. 9.1–9.23, Aug. 2008.

[35] S. Remarsu and S. Kundu, "On process variation tolerant low cost thermal sensor design in 32 nm CMOS technology," in *Proc. Great Lakes Symp. VLSI*, 2009, pp. 487–492.

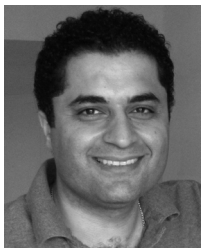
[36] Y. Zhang and A. Srivastava, "Accurate temperature estimation using noisy thermal sensors," in *Proc. Design Automat. Conf.*, 2009, pp. 472–477.

[37] R. Cochran and S. Reda, "Spectral techniques for high-resolution thermal characterization with limited sensor data," in *Proc. Design Automat. Conf.*, 2009, pp. 477–483.

[38] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first-generation CELL processor," in *Proc. Int. Solid-State Circuits Conf.*, 2007, pp. 184–185.

[39] R. Kuppuswamy, S. R. Sawant, S. Balasubramanian, P. Kaushik, N. Natarajan, and J. D. Gilbert, "Over one million TPCC with a 45 nm 6-Core Xeon™ CPU," in *Proc. Int. Solid-State Circuits Conf.*, 2009, pp. 70–71, 71a.

[40] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A tool for evaluating and synthesizing multimedia and communications systems," in *Proc. IEEE Micro.*, 1997, pp. 330–335.



Shervin Sharifi (S'10) received the B.S. degree from the Sharif University of Technology, Tehran, Iran, in 2000, and the M.S. degree from the University of Tehran, Tehran, in 2003, both in computer engineering. He is currently working toward the Ph.D. degree in computer engineering at the Department of Computer Science and Engineering, University of California, San Diego.

His current research interests include temperature and power aware system design.

Tajana Šimunić Rosing (M'01) received the M.S. degree in electrical engineering from the University of Arizona, Tucson, on the topics of high-speed interconnect and driver-receiver circuit design. She received the Ph.D. degree from Stanford University, Palo Alto, CA, in 2001, concurrently with finishing the Masters in engineering management on topic dynamic management of power consumption.

Prior to pursuing the Ph.D., she was a Senior Design Engineer with Altera Corporation, San Jose, CA. She was a Full Time Researcher with HP Labs, Palo Alto, CA, while working part-time at Stanford University. At Stanford, she led the research of a number of graduate students and has taught graduate level classes. She is currently an Assistant Professor with the Department of Computer Science and Engineering, University of California, San Diego. Her current research interests include energy efficient computing, embedded, and wireless systems.

Dr. Rosing has served on a number of technical paper committees, and is currently an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING.