Journal of Low Power Electronics
Vol. 12, 1–13, 2016

Online Inertia-Based Temperature Estimation for Reliability Enhancement

Mayank Chhablani[1,*], Israel Koren[2], and C. M. Krishna[2]

[1] *AMD, Inc., Boxborough, MA, 01719, USA*
[2] *Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, 01003, USA*

(Received: 15 April 2016; Accepted: 11 July 2016)

With the advent of technology scaling and the increased use of high performance multi-cores in life-critical applications, reliability has become an increasingly pressing issue. High utilization can cause localized thermal elevations (hotspots) which in turn, accelerates the aging of semiconductor devices. As a result, ensuring reliable operation of the processors has become a challenging task. A cost effective scheme for estimating on-chip temperature is crucial as it is needed for estimating the circuit reliability. In this work, we present a light-weight temperature estimation technique that is based on monitoring hardware events. Unlike previously proposed hardware counter-based approaches, our approach uses a linear-temporal-feedback estimator, taking into account the effect of thermal inertia. In simulation experiments, the proposed approach shows an average absolute error of less than 2.5 °C with standard deviation of <2 °C. Furthermore, if an on-chip temperature sensor is available, our modified technique can better tolerate ambient temperature variability. We then present a counter-based technique to estimate the Thermal Accelerated Aging Factor (TAAF), which is an indicator of lifetime reliability. Our results demonstrate that the estimation accuracy is adequate.

Keywords: Temperature Estimation, Reliability Estimation, Performance Counters, Localized Hotspot.

1. INTRODUCTION

Lifetime reliability has become a significant concern for system design because of the dramatic escalation in processor power densities over recent years. The occurrence of on-chip hotspots accelerates circuit wear-out, resulting in premature chip aging. It is therefore important to be able to accurately estimate on-chip temperatures so that steps can be taken to mitigate thermal damage. Ideally, such an estimation technique should not require substantial hardware support and should be very lightweight to employ, requiring no more than a handful (two or three) multiplications and additions.

In this paper, we present a temperature estimation technique that meets the aforementioned criteria. This approach takes into account both current activity as well as the residual temperature effects of prior activity. In order to be lightweight, we have focused on a linear function of temperature as a function of activity. Since the underlying phenomena driving temperature is a nonlinear function of activity, we subdivide activity levels into regions and obtain a separate linear estimate for each region, thereby obtaining a piecewise linear estimate. We also discuss how the impact of uncertainty in ambient temperature can be mitigated by augmenting our approach.

Collecting the thermal history of various parts of a chip is important since thermal damage is cumulative. A chip that has suffered considerable thermal stress in the past is more vulnerable to the effects of heating than another which has not. This can be quantified by the *Thermal Accelerated Aging Factor* (TAAF) which is the ratio of the effective (thermally accelerated) age of a circuit to its chronological age. As the TAAF is exponentially increasing with temperature, means for controlling the on-chip temperature are needed. Any procedure to mitigate thermal damage obviously relies on an accurate technique to estimate the on-chip temperature.

Our approach can be outlined as follows. Modern processors have a small number of performance counters which can monitor the activity of various functional units in the processor. By using standard feature-selection techniques from artificial intelligence, we identify an appropriate set of events to monitor. The idea is to select a subset of events that are highly correlated with the temperature but

[*] Author to whom correspondence should be addressed.
Email: Mayank.Chhablani@amd.com

are relatively uncorrelated to each other. By this means, we increase the effective information gleaned from a limited number of performance counters. Heat-flow models are then used to obtain a regression formula linking the previous temperature and counter values to the present temperature.

This paper is organized as follows. In Section 2, we present relevant technical background. Section 3 provides a brief summary of prior work in thermal estimation, including the use of performance counters. Section 4 describes our approach and Section 5 explains our experimental framework; this is followed by detailed illustrative numerical results in Section 6. Section 7 concludes the paper.

2. TECHNICAL BACKGROUND

The principal causes of processor failure (including negative bias temperature instability, time-dependent dielectric breakdown, stress migration, electro-migration, and hot carrier injection) occur at rates that exponentially increase with temperature. This must be ameliorated by mechanisms such as clock and power gating, voltage and frequency scaling, or throttling of selected units.[1] In order to trigger such control mechanisms, we need a lightweight and accurate temperature estimator. With run-time knowledge of temperature, a processor can adapt its operation to improve lifetime reliability.

Several current processors incorporate a small number of thermal sensors to monitor their temperature. Digital Thermal Sensors (DTS) have been incorporated into several Intel and AMD CPU families, but software access is restricted to only core temperature registers.[2] Intel's Sandybridge and AMD's Quad-Core Opteron incorporate 12 and 38 thermal sensors, respectively. This approach does have several drawbacks:

(1) The area of the sensor has to be large to provide high precision,

(2) Sensors measure the average temperature of the core which could miss localized hotspots, and

(3) Determining the number of these sensors, their calibration and placement are complicated as hotspots move over time.[3]

Due to these limitations of thermal sensors, we need an augmented approach for thermal sensing. On-chip performance counters present an attractive alternative or supplement to thermal sensors.

Performance counters are already available in today's high-end microprocessors for debugging and performance characterization. These counters are used to monitor certain events and activity levels like L1-cache hits/misses, functional unit accesses and branch-mispredictions. The number of events captured by the performance counters varies across processor families and their implementation. There are limitations on how many events can be simultaneously measured.[3] For example, AMD Athlon64, Opteron, and Phenom processors provide four performance counters to measure hardware events occurring during program execution. Intel's SandyBridge has 3 fixed counters, 4 general-purpose counters and 4 RAPL energy counters. The RAPL energy counters monitor *maximum average* power. As a result, these counters may not be able to catch the localized thermal elevation events pertaining to specific block(s). Therefore, we have chosen performance counters to estimate the temperature of the hottest blocks (since thermal damage increases exponentially with temperature, it is the hottest blocks that one is most concerned about).

3. REVIEW OF RELATED WORK

As power densities have increased in recent years, ameliorating their thermal impact has received considerable attention from the research community. Below we summarize some recent approaches to estimate power consumption and temperature.

3.1. Monitoring Power Consumption

Various works have used performance counters for power-monitoring. In Ref. [4], Singh et al. developed a linear model for power, on a per-unit basis. Their methodology uses a subset of performance counters, based on their correlation with power consumption, and performs linear regression using the Ordinary Least Square (OLS) estimator. Four different policies are suggested to take corrective measures when the power envelope is breached.

In Ref. [5], Isci and Martonosi proposed an online power estimation and synchronization model. Per-unit power measurement was done by sampling performance counters at fine-grained cycle granularity. In Ref. [6], Rodrigues et al. have shown that three performance counters are sufficient to estimate the dynamic power consumption of processors with 95% accuracy. In Ref. [7], Yang et al. proposed an application-specific design flow for soft-error reliability optimization and energy-efficiency in tandem. Although power-aware-techniques can gauge the potential breach of the power budget/envelope, these techniques do not account for the thermal impact of increasing localized hotspots which degrade the performance and life span of a processor.

3.2. Thermal-Aware-Techniques

In Ref. [8] a detailed thermal model, HotSpot, is described which represents the architectural blocks of a chip as an equivalent network of thermal resistances and capacitances with the power consumed by each functional unit being modeled as a current source. The temperature difference between two points is analogous to voltage and the resulting heat flow is analogous to current flow. Thermal capacitance measures the amount of heat energy required to raise the temperature by one degree; thermal resistance measures the amount of heat flow resulting from a unit difference in temperature.

Hotspot has three heat flow models. The lateral model expresses the flow of heat between a sub-unit and its neighbors. Two vertical models are provided to take into account the vertical effluxion of heat to the heat spreader, the heat sink and the on-chip interconnect and substrate.

3.2.1. Online-Thermal Modeling Using Sensors

Many researchers have assumed the existence of thermal sensors for monitoring core temperature.[9] In Ref. [11], thermal samples are collected from on-board sensors, and then temperature-abatement procedures are triggered when appropriate.

In Ref. [12], Coskun et al. suggested a proactive temperature measurement technique using an Autoregressive Moving Average (ARMA) technique. In this technique, a temperature trace of the current workload is collected from thermal sensors. This trace is sent to a run-time predictor so as to monitor the change in thermal characteristics due to the dynamic nature of the workload.

Validation of this model, which takes place at run-time, incurs overhead since it requires computation of differential residuals. Moreover, thermal sensors come with a few limitations: they may be too expensive and their placement is not trivial as hotspots move around during execution.[13]

Although sensors have known limitations, the technique in Ref. [11] can be used in synergy with our augmented methodology to mitigate the impact of uncertainty in ambient temperature as discussed in Section 6.4.

3.2.2. Online-Thermal Modeling Using Hotspot

In Ref. [14], Bao et al. proposed an online thermal and power estimation that employs Hotspot and uses an iterative process. This procedure assumes an initial temperature and converges to a temperature estimate. Lee and Skadron augmented the Hotspot model to capture micro-architectural events using performance counters.[13] They employed the power model from Ref. [5] and integrated it with Hotspot in real-time. They have used it to capture the variation in the gradient of maximum temperature across functional units.

Merkel and Bellosa suggested a hybrid approach using task activity vectors; they sampled the utilization from performance counters every millisecond and estimated power;[15] Hotspot was then used to estimate the temperature. However, as the Hotspot model requires solving differential equations, it turned out to be an expensive solution for temperature monitoring in practice.

3.2.3. Offline-Modeling [Using Micro-Architectural Events/Performance Counters]

Reading temperature data from thermal sensors or incorporating Hotspot[8] or TILTS[16] at run-time seems to be a plausible direction for thermal estimations but these methods have the above mentioned drawbacks. Chung and Skadron proposed a fine-grain localized temperature estimation technique using on-chip events. They showed that by sampling performance counters at fine granularity and then using regression analysis, the temperature trace can be estimated.[3] They employed similar metrics to those in Ref. [5] and used Hotspot to estimate temperature. Lee et al. predicted the localized temperature of a target functional unit, using performance counters, and performed Dynamic Voltage and Frequency Scaling (DVFS), with the help of a linear regression analysis.[17]

It is important to note two aspects of the aforementioned work. First, they used a two-step process to estimate the reference temperature, i.e., first the power is estimated from the performance counters and then the temperature is estimated using Hotspot, which may incur some error in the reference temperature itself. Second, their linear expression contains a large offset, which may result in overestimation when there is no activity.

In Ref. [3], the authors have noted that the past thermal traces do not contribute towards estimating the patterns of thermal fluctuations at run-time. More recently, Upton and Hazelwood argued that linear regression is not a suitable choice for modeling full-core temperature based on performance counters and instruction stream (i.e., instruction category).[18] They also indicated that the inclusion of temperature history tends to result in overweighting of the past temperature, leading to an increase in estimation error.

However, one key point to note is that, temperature gradually changes from one program phase to the other. In addition, although[18] has shown that the average error was significantly reduced, averaging out arithmetically can be skewed by outliers, which may result in an inaccurate estimation.

In contrast, our temperature estimation technique depends on the previous thermal history that proved to have a significant impact on the accuracy of the thermal estimation because temperature changes gradually due to thermal inertia.[a] Including the thermal history also results in a relatively small offset in the estimator, which not only mitigates the overestimation during no-activity phases, but also enables efficient reliability-aware design and decisions.[19]

4. ESTIMATION METHODOLOGY

Our methodology consists of the following steps:

(1) Given a constraint on the number of performance counters, determine the most appropriate events to be counted.

(2) Use regression together with heat-flow simulations to obtain a temperature estimate of each unit as a linear function of counter values and the previous temperature estimate.

[a]The thermal inertia in this context can be defined as rate at which the temperature of chip approaches the ambient temperature if there is no power input or a steady-state if there is a steady power inflow.

(3) Use k-fold cross-validation to evaluate the accuracy of this approach. If the accuracy needs to be improved, divide the range of activity into multiple sub-ranges and repeat Step 2 to obtain a linear function for each of the sub-ranges. We now describe each of these steps in detail.

4.1. Correlation-Based Feature Subset Selection

Today's processors have a limit on the number of activities that can be simultaneously monitored with counters. We must therefore carefully select those which, when monitored together, contribute the greatest amount of information towards temperature estimation. For this purpose, we use the well-known correlation-based subset selection approach of Hall.[20] A best-fit approach is used along with backtracking to incrementally add counters which are highly correlated with temperature and only lightly correlated with each other (increasing the value of the information derived from each by reducing information redundancy).

Once the counters have been selected, the sampling frequency has to be determined. Too high a frequency will result in frequent interruptions and an increased overhead; too low a frequency will result in some transient temperature dips and peaks being missed. We can use thermal capacitance and resistance information to guide us; the product of these quantities gives us a thermal time constant, which is a measure of the thermal inertia of the chip. The thermal time constant is typically between 5 milliseconds and 300 milliseconds (ms).[21] As long as the sampling frequency is greater than the inverse of the thermal time constant, the update rate should be sufficient. We must, of course, validate any such choice by means of thermal simulation.

Testing is done using k-fold cross validation.[22] In this technique, the data are randomly split into k mutually exclusive subsets (folds) of approximate equal size with one of these serving as a training set. Correlation-based feature subset selection is then performed based on this information. Once obtained, we can then test the effectiveness of the subset selection on the remaining $k-1$ subsets. This can be repeated k times, each time taking a different one of the k subsets as the training set and the others as the test sets.

4.2. Inertia-Based Linear Regression

We derive a formula to estimate the current temperature as a linear function of counter values and the previous temperature estimate. In particular, for every unit u of the chip, the estimated temperature in the n-th sampling instant is

$$T^u_{\text{est}}(n) = \underbrace{\sum_i \alpha_i C_i(n)}_{X} + \underbrace{\beta T^u_{\text{est}}(n-1)}_{Y} + \underbrace{\delta}_{Z} \qquad (1)$$

where $C_i(n)$ is the value of the i-th counter at the n-th sampling instant. In the equation above, the term X represents the impact of energy consumption in the just-expired time slice, Y is the impact of heat carried over from the past, and Z is the impact of leakage and other effects. The values of α_i, β and δ are obtained through standard linear regression.

There are instances where estimation accuracy can be enhanced by dividing the activity region into several subregions and deriving a separate linear formula for estimating the temperature in each subregion. In particular, we have in certain cases found it useful to divide into two subregions: high-activity and normal-activity, and to carry out a separate regression for each of these subregions. This forms the basis of our *History-induced Dual Estimator*. Doing so does not greatly add to the runtime complexity of making an estimate: selection of the appropriate estimation expression can be carried out rapidly and storing the additional expression coefficients in a lookup table does not take much memory. In practice, a workload-manager can trigger thermal-abatement techniques at run-time by probing (or, sampling) performance counters and employing the desired estimator.

4.3. History-induced Dual Estimator

As the name suggests History-induced Dual Estimator (HiDE) is a technique wherein we switch from one expression to other at run-time, based on the current activity of the unit. We have chosen Integer Register Access Per Cycle (IRA) and Number Of Floating-Point Instructions Per Cycle (NFP) as the base activity (b_{act}) for determining the threshold at which estimators will switch for Integer unit-variants and Floating point unit-variants, respectively. The two expressions will be generated offline and will be based on low and high base activity of the unit, yielding a High Activity Estimator (*ha_est*) and a Low Activity Estimator (*la_est*), as shown below.

(2)

where the first expression corresponds to *la_est*, the second to *ha_est* and τ represents the switching threshold. Also, the subset of performance counters for *la_est* may be different from that for *ha_est*. The choice of τ for these expressions is crucial because if this threshold is not chosen properly, estimators will be generated from a skewed thermal distribution and may result in an inaccurate estimation. Moreover, this threshold needs to be determined for each unit under consideration because of different base activity, (b_{act}), taken into account. Since this is done offline and only once, it has no implication on the runtime overhead.

5. EXPERIMENTAL ENVIRONMENT

In this section, we describe our simulation framework: see Figure 1. The system behavior at cycle-level is simulated using gem5[23] and the sampling of the performance counters is done at a granularity of 20 million cycles (10 ms) for AMD Athlon 64 at 65 nm[24] and the Alpha

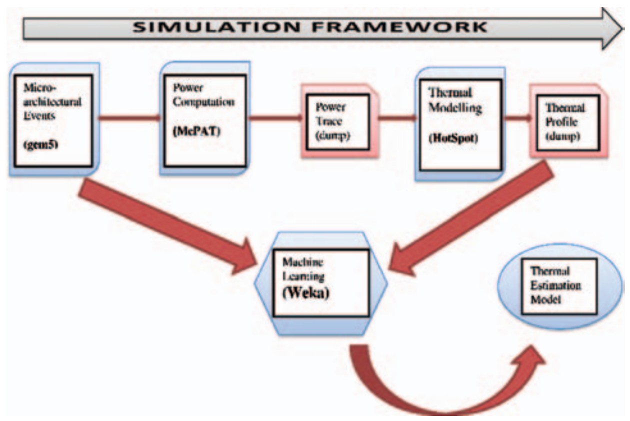

Fig. 1. Simulation framework.

21264 processor core at 90 nm technology node, operating at 2 GHz. Table I shows the benchmarks that we used as representative subsets from the MiBench[25] and SPEC Benchmark suites.[26] MiBench consists of embedded system applications while SPEC contains general-purpose computing applications. The selected benchmarks (see Table I) were chosen not only based on instructions types (e.g., integer-intensive and floating-point intensive) but also on the basis of thermal characteristics (e.g., slow and fast changing and gradually increasing thermal fluctuations). We have tested our estimator on individual suites and mixed suites, along with different training and testing sets. To model periodic tasks, we have employed these benchmarks with repetition; each of these benchmarks were executed for 4 billion cycles, after fast-forwarding past 1 billion cycles.

In order to focus on a representative set of events which significantly correlate with thermal variations, we have examined twenty four performance counters which are as follows.

• *Instructions/Micro-Operations Per Cycle* (*IPC*): IPC can be important to monitor because the number of instructions per cycle has a direct correlation with temperature due to the unit's activity. In case of AMD Athlon, we focused on micro-ops per cycle instead of on X86 complex instructions.

• *Functional Unit Access* (*FUA*): The accesses to a particular unit plays an important role in projecting the thermal fluctuations in that unit. We explored the following performance counters related to FUA: IntRegAccess (IRA), FPRegAccess (FPRA), IntMapAccess (IMA), FPMapAccess (FPMA), IntQAccess (IQA), IntExecAccess (IEA), FPUnitAccess (FPUA), BPredAccess (BPUA), IssueRate.

• *Dispatch Stalls* (*D_Stalls*): Here, dispatch stalls include stalls due to re-order buffer (ROB), load store queues (LSQ), reservation stations (RS), register map and register alias table (RAT). This counter has negative correlation with temperature, because while experiencing dependencies, activity in a unit decreases, thereby resulting in a gradual fall in temperature.

• *Hits/Miss Counters*: The following counters can be grouped under this category: L2Misses (L2m), L1Hits (L1h), L1Misses (L1m), L2Hits (L2h). These counters also have a significant impact on power consumption. Hence, we have considered these as secondary counters while estimating thermal fluctuations.

• *Fetch and Speculative Counters*: Many instructions are speculatively executed in the pipeline and may need to be flushed due to execution on a false (mispredicted) path. Therefore, execution of these instructions and pipeline flushing plays a significant role in thermal estimation. The counters explored under this category are: Number of Floating Point Instructions (NFP), Number of Fetched Instructions (Fetch_Insts), Branch Correctly Predicted (BCP), Branch Mis-predictions (BMP), Load-Store Instructions (LSInsts) and Branch Instructions (BrInsts).

The Correlation-Based Subset Selection (*CfsSubsetEval*) method,[20] as described in Section 4, was then applied to select a limited subset of performance counters for thermal estimation.

Floating-point units such as Floating-Point Queue (FPQ), Floating-Point Map (FPMap) and Floating-Point Register (FPReg) can be grouped together, since they show very similar thermal behavior, and hence the behavior of these units can be estimated with the same performance counters. Therefore, we have merged FPQ, FPMap and FPReg unit into one single unit, thus obtaining a simplified floorplan of Alpha 21264 as shown in Figure 2(b). While generating the simplified floorplan, using *Hotfloorplan*,[8] we aggregated the power of those units which were merged to *FPReg unit* in the new floorplan.

We then integrated gem5 with a power modeling tool, McPAT;[27] both dynamic (switching) and static (leakage) power are accounted for by this tool. McPAT allows one to get the area specifications of individual functional units, which is helpful in thermal modeling. Power traces for each unit (from McPAT) drive the thermal model of HotSpot.[8]

In our experiments we have considered temperature from Hotspot as the reference temperature. Table II summarizes the modified configuration parameters[b] for

Table I. Selected benchmarks.

Suite	Benchmarks
MiBench	Adpcm, basicmath, bitcount, blowfish (encode and decode), crc, dijkstra, fft, fft-inverse, gsm (encode and decode), jpeg (encode and decode), lame, patricia, qsort, rijndael-decode, sha, susan, typeset
SPEC2006	Astar, bwaves, bzip2, calculix, dealII, gcc, h264ref, hmmer, mcf, namd, soplex, wrf
SPEC2000	Ammp, applu, art, equake, mesa, mgrid

[b]We set the Hotspot's[28] parameters to fit the thermal characteristics of single core die.

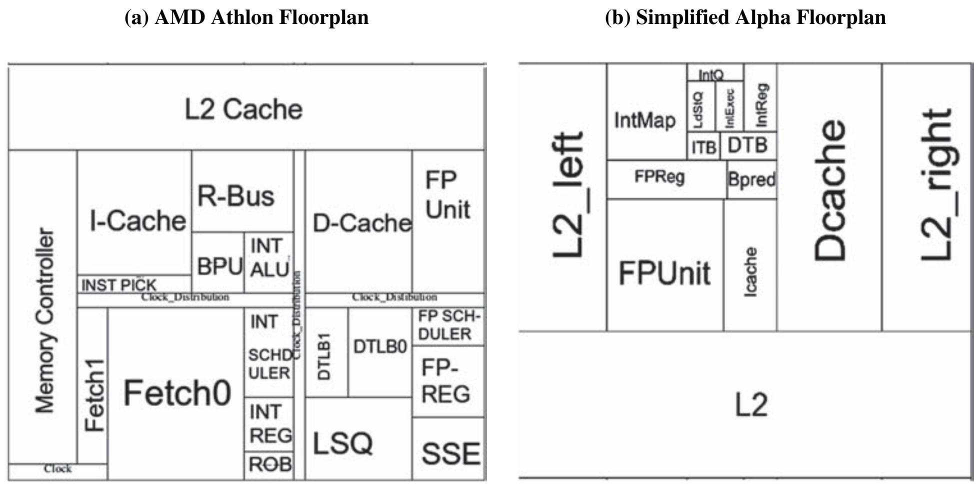

Fig. 2. AMD Athlon64 (65 nm) and alpha 21264 Core (90 nm) floorplans.

Table II. HotSpot configuration parameters.

HotSpot parameter	Value
Chip thickness	0.15 mm
Core area	241.883 mm^2
Convection capacitance	140.4 J K^{-1}
Convection resistance	0.7 K W^{-1}
Heat sink side	0.0526 m
Heat spreader side	0.026 m
Substrate side	0.02 m
Ambient temperature	45 °C

HotSpot. The temperature model requires specification of the processor floorplan; we extracted the area specifications from McPAT and used Hotfloorplan[8, c] for this purpose. Once we generate the full thermal trace, we use a software tool, Weka,[29] which has a time-series framework[30] for generating thermal estimators, based on linear regression.

Note that the accuracy of the Hotspot model has been extensively validated using power-thermal maps;[8] it can therefore be used with confidence as representing the actual temperature.

5.1. Time-Series Forecasting Model

Weka provides a *Time Series Forecasting Configuration* in which, a time-dependent series of observable variables (in our case, estimated temperature trace history and on-chip events) can lead to the development of an estimation expression using regression. We have chosen linear regression as the base learner for the estimation technique.

Weka's time series framework follows a machine learning approach to model a time-series. It encodes input activity data with time dependency via additional input fields, representing, in our case, historical thermal fluctuations. This process is called *flattening*.

Once we have flattened our observable input variables, i.e., performance counters and temperature history, we can apply Multiple Linear Regression (MLR) with the M5 Descriptor selection method, with the objective of minimizing the sum of squared residuals. The M5 Descriptor selection method is a process by which the features (performance counters, in our case) with the smallest regression coefficients are stepwise removed from the model until no reduction is observed in the average estimation error as given by Akaike Information Criteria (AIC).[30] The AIC not only rewards goodness of fit, but also includes a penalty that is an increasing function of the number of estimated parameters. This penalty discourages over-fitting.

After applying MLR with the M5 Descriptor selection method, an estimation expression is generated as explained in Section 4. Clearly, the full trace of activity is not available to an actual system during runtime. Instead, we have to select the parameters (i.e., weights in Eq. (1)) based on an offline study of a training set. Note that in an embedded system, the task set is known in advance (even if the rate of invocation of individual tasks may vary and the actual input data values are not known in advance).

6. EXPERIMENTAL RESULTS

We have evaluated our approach on three different benchmark subsets, MiBench, SPEC[d] and a mix of both MiBench and SPEC suites. Since the input data are not known in advance, we have trained our technique on one input set and tested it on an entirely different input set. While presenting our experimental results we will also address the following issues. First, what is an appropriate selection of the sampling rate of the counter values? Any approach that requires an excessively high sampling rate for accurate estimation will impose a noticeable overhead.

[c]HotFloorplan is a microarchitecture level thermal-aware floorplanning tool which makes use of simulated annealing for slicing floorplans.

[d]SPEC encompasses both the SPEC2000 and SPEC2006 benchmark suites.

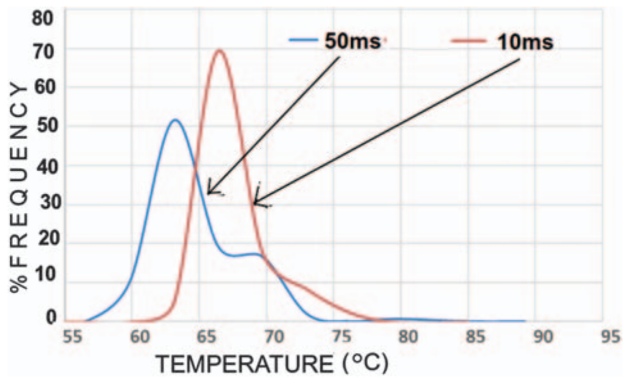

Fig. 3. Sampling window: sampling at rates of 10 ms and 50 ms.

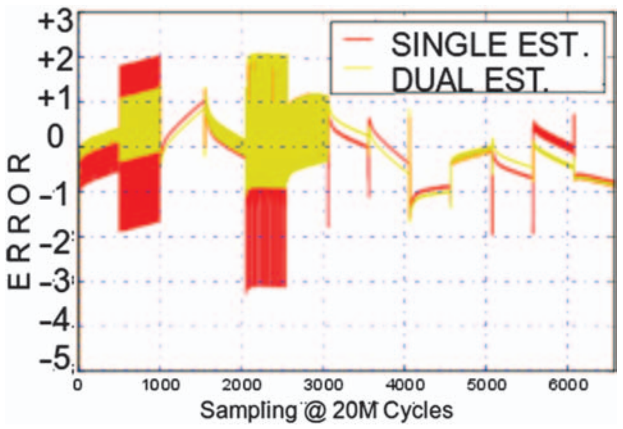

Fig. 4. Temperature error variation for dual and single estimators in Table III.

Second, how much improvement in accuracy can we get by dividing the activity region into subregions and having a separate expression for each? Third, how does our approach compare with prior work? Fourth, what is the impact of uncertainty in the ambient temperature? Fifth, if we have a temperature sensor in addition to counter values, what impact does this have on the temperature estimation accuracy? Sixth, how beneficial is it to estimate TAAF (which is a proxy for lifetime reliability) directly using performance counters?

6.1. Selection of Sampling Rate

Experiments were carried out to determine the appropriate performance counter sampling frequency. Figure 3 shows the impact of the sampling window on the thermal characteristics of a set of benchmarks from Mibench and SPEC suites viz. *bitcount, namd, soplex, calculix* and *bwaves*.

We found that sampling periods of less than 10 ms (equivalent to 20 Mcycles) produced near-identical results; beyond that, however, loss of fidelity was evident; for example, Figure 3 compares 20 MCycle and 100 MCycle (or, 50 ms) periods. It is evident from Figure 3 that at 50 ms sampling intervals, we are missing several high temperature peaks.

This is also justified by our processor thermal time constant which is 20 ms. Based on such data, we selected a sampling window of 20 million cycles, which is equivalent to a sampling interval of 10 ms on a 2 GHz processor. Such an interval provides a natural opportunity for software to read on-chip events as it is within the range of sampling granularity of commercial operating systems.[3]

6.2. Integer Unit(s) Temperature Estimation

The integer scheduler and integer register file are usually some of the hottest units in the core,[24, 31] resulting in a hotspot of up to 92.4 °C, unless dynamic thermal management (DTM) is employed. We therefore, focus on these units for our illustration here; tracking of the other units follows the same approach.

We used the technique described earlier to construct estimators for the set of workloads listed in Table III. The level of activity was measured by counting the IssueRate per cycle for the Integer Scheduler unit and the number of Integer Register Accesses (IRA) per cycle in case of the IntReg unit. *Our experiments have shown that these selected performance counters exhibiting a higher than 0.97 correlation with temperature, do not vary greatly across technologies and architectures.*

Two approaches are compared. In the first, a single regression expression is derived for the entire range of activity. In the second, a threshold, τ (IRA), is defined and two expressions are derived, one for below-threshold and the other for above-threshold activity. Two candidates for threshold were considered: the mean and median of the observed IRA.

Figure 4 compares the single- and dual-estimator (i.e., HiDE) approaches that use the expressions shown in Table III. The dual estimator, on an average, provides 1.2 °C better estimation accuracy than the corresponding single estimator.

Table III. [Alpha21264 Core] MiBench workloads and estimation expressions when different inputs sets were considered.

	Dual estimator (τ = Median = 2.1)		
Workloads	Below median	Above median	Single estimator
bitcount, susan, jpeg-decode, lame, patricia, dijkstra, rijndael-encode, sha, adpcm, crc, fft, fft-inverse, gsm-encode	$T_{\text{est}}^{\text{IntReg}}(n) = 3.56 * \text{IRA} + 0.18 * T_{\text{est}}^{\text{IntReg}}(n-1) + 48.65$	$T_{\text{est}}^{\text{IntReg}}(n) = 2.70 * \text{IRA} + 0.46 * T_{\text{est}}^{\text{IntReg}}(n-1) + 32.75$	$T_{\text{est}}^{\text{IntReg}}(n) = 3.55 * \text{IRA} + 0.20 * T_{\text{est}}^{\text{IntReg}}(n-1) + 50.92$

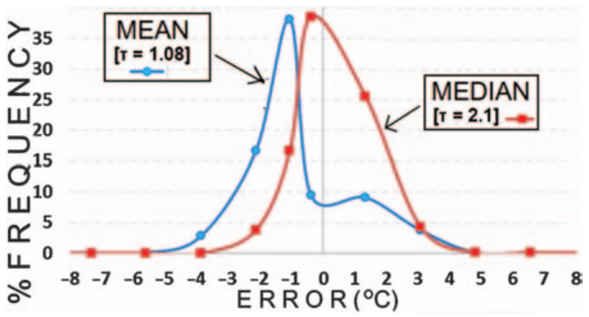

Fig. 5. Threshold selection for the dual estimator in Table III.

As it is insensitive to the value of large outliers, the median is likely to be a better choice of threshold than the mean when using a dual estimator; this intuition is validated by the results shown in Figure 5. We also tested our approach using distinct sets for training (i.e., obtaining the estimator expression) and testing (calculating the estimation error). This allows us to check how transferable the estimator is, from one set of tasks to another.

Figure 6 shows the distribution of error for the worst-case testing set, illustrating the low error swing in the high temperature zone for MiBench and SPEC06 Suites. As evident, when used on an entirely different workload set, the estimator exhibits acceptably small absolute errors, mostly below 2.5 °C. Note that since thermal damage tends to increase exponentially with temperature, estimation accuracy is more important at high temperatures.

6.2.1. *Comparison with Previous (Baseline) Approach*

Similar experiments were carried out on the SPEC benchmark suite. Using these benchmarks we can compare the quality of our estimator against that developed in Ref. [3]. Table IV compares the dual estimator for our approach (for high and low activity) to the estimator derived using the approach in Ref. [3]. The latter resulted in error range of [−9.19 °C, +10.95 °C] for the *bzip* benchmark. By comparison, our approach has an error swing of [−1.52 °C, +2.59 °C].

Our experimental results have shown that the effective average error swing, in the high temperature zone,

Table IV. [AMD Athlon64 Core] comparative study (SPEC Suite): Estimation expression for the temperature of the IntReg unit for a given training and testing set.

Dual estimator ($\tau = 1.3$)		Baseline-estimator based on
Below median	Above median	Ref. [3]
$T_{\text{est}}^{\text{IntReg}}(n)$ $= 3.13 * IRA + 0.54$ $* T_{\text{est}}^{\text{IntReg}}(n-1)$ $+ 24.31$	$T_{\text{est}}^{\text{IntReg}}(n)$ $= 3.00 * IRA + 0.34$ $* T_{\text{est}}^{\text{IntReg}}(n-1)$ $+ 35.06$	$T_{\text{est}}^{\text{IntReg}}(n)$ $= 7.05 * IRA$ $+ 56.67$

is between −2 °C to +2 °C and between −4.5 °C to +2.5 °C in the moderate temperature zone. Note that at the beginning, the estimate starts with a large error of up to −16 °C but this error comes down in about 20 samples. The rationale behind this behavior is that, when applying linear regression the assumed initial condition is inaccurate but because of geometric progression of the estimator, the error phases out. Figure 7 shows how the initial error phases out and dynamic temperature changes being captured.

Figure 8 shows the overall error swing and average absolute estimation error across the Integer Register and Integer Scheduler units for the AMD Athlon. The overall effective error range across different training and testing sets is between −3.5 °C to +4.5 °C. Note also that our estimate of the peak temperature has an underestimate of about +2 °C at a peak (average) temperature of 84 °C.

Thus far, all the expressions have been linear. We have also experimented using non-linear expressions; however, the estimation error turned out to be much worse, approximately −15 °C. Also, we tested out the non-linear time series technique using Artificial Neural Networks[32] but this provided no improvement over the linear technique.

6.3. Floating-Point Unit Temperature Estimation

Another potential unit(s) which may need thermal monitoring is the Floating-Point Scheduler and Floating-Point Register (FPReg) units, where thermal variation can be up to 125 °C, if no thermal management is applied.[31] Similar to the IntReg unit, we have chosen a set of representative

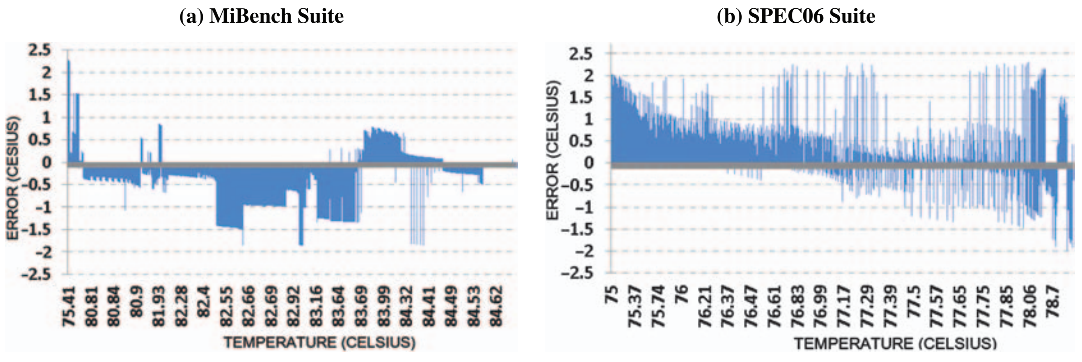

Fig. 6. Testing-set error distribution in the high temperature region (>75 °C).

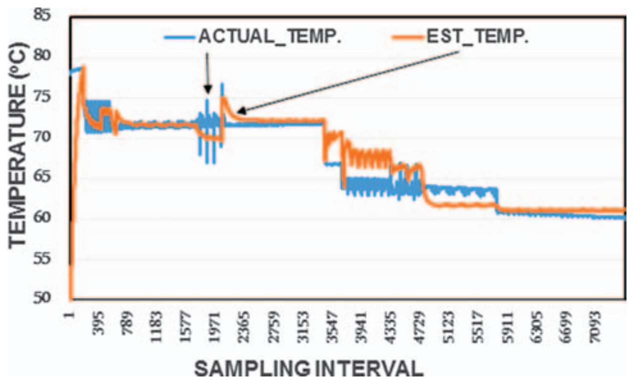

Fig. 7. Capturing of dynamic temperature changes by our estimation methodology for IntReg unit in AMD-Athlon processor.

benchmarks for the estimation (training and testing) of the temperature of the FPReg unit. The benchmarks chosen are: *bwaves, milc, namd, dealII, soplex, calculix, mgrid, applu, mesa, art, equake* and *ammp*. Here, we have used the Number of Floating-Point Instructions per cycle (NFP) as the representative performance counter for determining the threshold (τ) for switching between estimators. However, unlike IntReg, here, the dual estimator provided only a small improvement over the single estimator and hence, we decided to follow the single estimator approach for the thermal estimation of FPReg. The thermal estimator for FPReg Unit is as follows.

$$T_{\text{est}}^{\text{FPReg}}(n) = 1.71 * \text{NFP} + 12.61 * \text{BMP} + 0.92 * T_{\text{est}}^{\text{FPReg}}(n-1) + 4.58 \qquad (3)$$

Similar experiments were carried out with respect to FPScheduler and the derived estimator for the MiBench Suite is:

$$T_{\text{est}}^{\text{FPSched}}(n) = 0.15 * \text{IssueRate} + 0.66 * T_{\text{est}}^{\text{FPSched}}(n-1) + 21.08 \qquad (4)$$

Figure 9 shows the overall error variation across different floating-point units. The error swing is in the range [−2.8 °C, +2 °C], demonstrating the robustness of our approach.

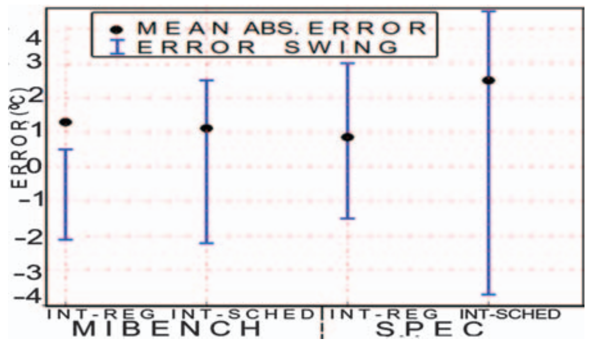

Fig. 8. Overall error statistics across different training-testing sets for the hottest integer units.

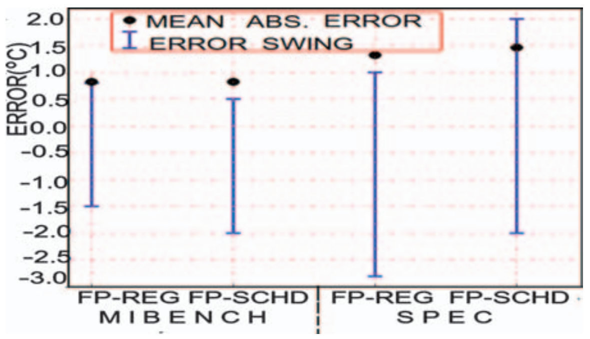

Fig. 9. Overall error variation across the floating-point units for AMD Athlon.

6.4. Impact of Uncertainty in Ambient Temperature

The estimation accuracy which we have reported above was obtained assuming the ambient temperature was known. However, in practice the ambient temperature may be different from the one on the basis of which the estimator was generated.

Figure 10 shows a linear increase in the offset term of the FPReg temperature estimator with respect to the assumed ambient temperature. The impact of variability in the ambient temperature is evident from Figure 11, which shows a large underestimation (in the FPReg temperature) of −20 °C, when mixed workloads are used. Therefore, in order to mitigate this issue, we have considered a design that includes an on-chip temperature sensor. Such a sensor will clearly reduce the impact of the variability in the assumed ambient temperature. However, on-line sensors have their own inaccuracies and the question is whether the use of performance counters can yield a robust estimator that can tolerate the ambient temperature variability.

6.4.1. Impact of an On-Chip Sensor

Table V shows INTScheduler temperature estimators (for SPEC06 Suites) when an on-chip thermal sensor has been integrated into the chip. The estimator is generated by combining thermal traces over a range of ambient temperatures, 20 °C to 60 °C, with an interval of 10 °C. In addition, we have introduced an inaccuracy of [−4 °C, +4 °C] in

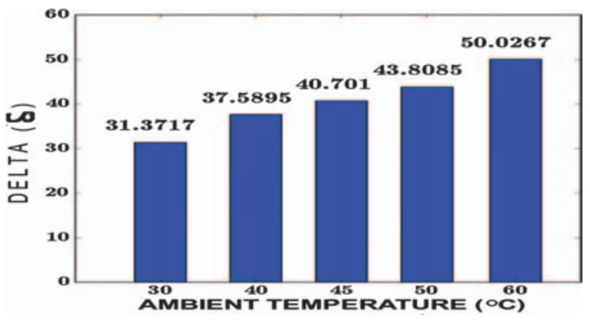

Fig. 10. MiBench suites: the offset (in the FPReg estimated temperature) as a function of the assumed ambient temperature.

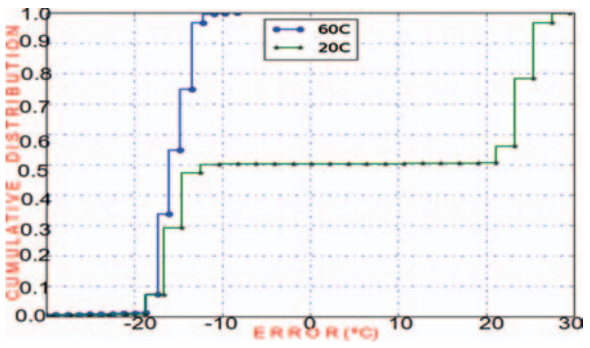

Fig. 11. Effect of uncertainty in ambient temperature: FPReg temperature estimation error when the estimator was trained for an assumed ambient temperature of 45 °C and tested on ambient temperatures of 20 °C and 60 °C.

the thermal sensor[33] reading, so as to simulate a practical scenario. The injected inaccuracy is linear which is based on the 2-point calibration method discussed in Ref. [34] and the sensor reading is obtained as per the following equation.

$$\text{Sensor\_Response} = 0.837 * \text{Actual\_Temperature} + 11.34 \tag{5}$$

Figure 12 shows the variation in the INTScheduler temperature estimation error for one testing set. As evident, our approach performs fairly well over thermal sensors; thereby having better accuracy in estimation by almost +1 °C. The overall estimation error across different training-testing sets for the selected range of ambient temperatures is within the tolerable range of [−3 °C, +4.59 °C].

Table VI shows the performance counter(s) chosen by the machine learning approach for potential hottest functional unit(s). Our experiments have shown that, while training the estimator, we need to train on the entire range of ambient temperatures and not just the outliers, otherwise the estimator may not be able to adequately tolerate the ambient temperature variability.

Having such an enhanced and effective temperature estimation scheme, it is worth evaluating the impact of our approach on the estimation of lifetime reliability. The following section demonstrates the efficacy of our approach on TAAF, a proxy for lifetime reliability, by deriving the similar methodology as explained in Section 4.

Table V. [AMD Athlon64 core] estimators for the intscheduler unit based on issuerate and the normalised on-chip sensor reading for the intscheduler, SR_SCHD.

Sensor type	Low activity estimator	High activity estimator
On-chip sensor	$T_{\text{est}}^{\text{IntSched}}(n) = 0.12 * \text{IssueRate} + 1.77 * SR\_SCHD + 0.98 * T_{\text{est}}^{\text{IntSched}}(n-1) + 0.01$	$T_{\text{est}}^{\text{IntSched}}(n) = 0.20 * \text{IssueRate} + 2.80 * SR\_SCHD + 0.97 * T_{\text{est}}^{\text{IntSched}}(n-1) - 0.18$

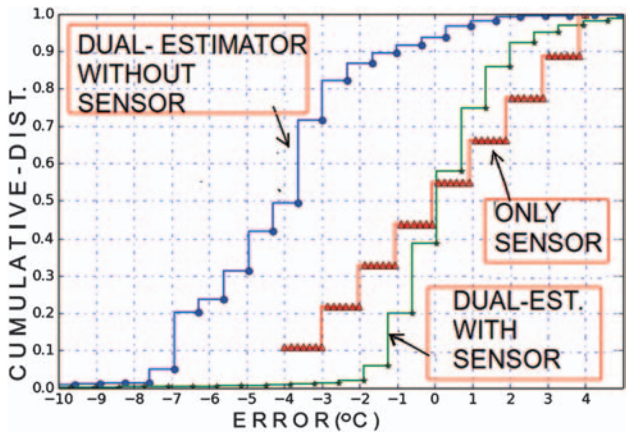

Fig. 12. SPEC suites: IntScheduler unit (AMD Athlon) temperature error variation for one of the testing sets, when the estimator was trained on a range of ambient temperatures (20 °C–60 °C) and tested on an ambient of 50 °C.

6.5. Thermal Accelerated Aging (TAAF)/Lifetime Reliability Estimation

In the past few years, instead of treating reliability improvement as an indirect benefit of thermal management, researchers have started to consider the reliability of VLSI circuit as an optimization objective. In Refs. [7, 19, 35] the authors have focused on reliability-aware-design and presented reliability-aware dynamic scheduling techniques and models. These previous works have suggested a potential for large energy savings and meaningful improvements in reliability. The basis of these techniques is the correct estimation of the failure rate, that is, the frequency at which a processor or its component fails. We use thermal accelerated aging (TAAF) as a proxy for estimating the impact on reliability. TAAF is defined for electromigration phenomenon as:[10]

$$\text{TAAF} = \frac{\text{MTTF (amb)}}{\text{MTTF }(n)} = \exp[(E_a/\kappa) * (1/T_{\text{amb}} - 1/T(n))] \tag{6}$$

where, *MTTF(amb)* is the Mean-Time-To-Failure at the ambient temperature, *MTTF*(n) is the Mean-Time-To-Failure corresponding to the estimated temperature at an instance n, E_a is activation energy, κ is the Boltzmann constant, T_{amb} is the ambient temperature in Kelvin, and $T(n)$ is the temperature at instance n in Kelvin. For copper, $E_a = 0.9$ eV.

Table VI. The performance counters chosen by machine-learning approach for different functional units.

INTScheduler	INTReg file	FPScheduler	FPUnit
IssueRate	Integer register accesses (IRA)	IssueRate	Number of floating point instructions (NFP) and number of branch mispredicts (BMP)

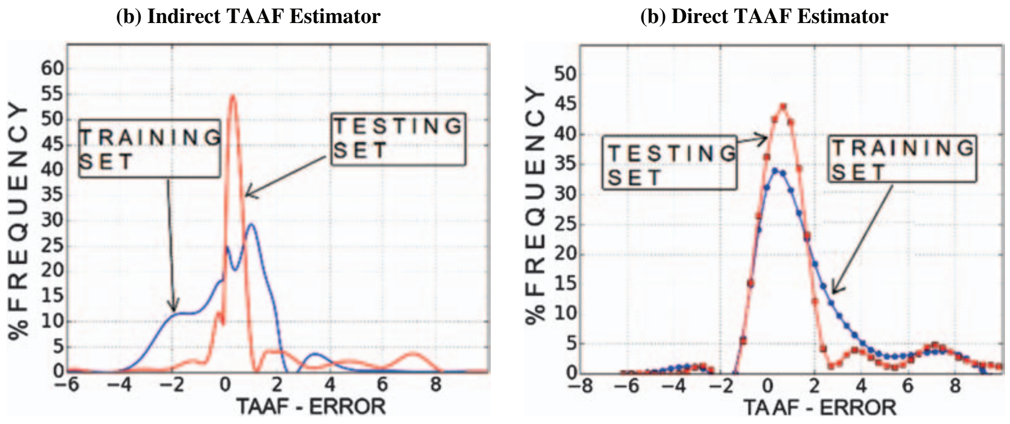

Fig. 13. [SPEC suites-AMD Athlon] TAAF estimator using (a) estimated temperature (b) direct TAAF monitoring using performance counters and accrued thermal stress.

The estimation of reliability at run-time is necessary for reliability aware scheduling and architecture design. Using Eq. (6) we monitor the TAAF, by back-tracking from the estimated temperature, as shown in Section 6.2. Figure 13(a) shows the estimation error in TAAF. As evident, the estimation error is in the range of [−4, +4].

We also made an attempt to estimate TAAF directly using performance counters. We followed the same methodology as explained in Section 4, replacing $T(n)$ with *TAAF*(n).

$$\begin{aligned}\text{TAAF}(n) &= 0.22 * \text{IssueRate} + 0.97\\ &\quad * \text{TAAF}(n-1) + 0.05 \end{aligned} \tag{7}$$

Equation (7) shows the TAAF estimator which takes into account the impact of the past thermal stress, i.e., TAAF($n-1$). We assumed the initial condition TAAF1 = 1, i.e., the state of no thermal acceleration. Figure 13(b) shows that the estimation error in TAAF is within the range [−2, +5].

Although estimating TAAF directly from performance counters is fast (Eq. (7)) but, it is somewhat more conservative compared to the indirect approach (i.e., back-tracking from estimated temperature). There is an overestimation associated with direct estimation, indicating faster thermal acceleration, which may force one to take conservative reliability-aware decisions. Still, both the indirect and direct estimations have an acceptable error level and selecting between them will depend on the system design needs. We also tried to incorporate sensor reading into the TAAF estimation for the Integer Scheduler unit. But, there was no significant improvement in estimation accuracy by including sensor reading.

We did similar experiments for the IntReg and FPReg units. Table VII shows TAAF estimators which were generated using the same methodology as used before for temperature estimation. Figure 14 shows the overall error swing, mean absolute error and standard deviation for the worst case testing set across two different floorplans, for the IntReg and FPReg units. As evident, the overall error is within a tolerable range of [−3.2, +5.2], with an average absolute error of +3. A better understanding of this TAAF error can be gained by looking into Figure 15, where actual instantaneous variation of TAAF is compared with estimated TAAF for one of the testing set.

This same notion can be extended to the FPScheduler unit, which becomes the hottest unit in case of floating-point intensive workloads. We also tried non-linear estimators but those estimators did not provide improvement over the linear (feedback) technique.

Table VII. [SPEC suites] TAAF estimators for the IntReg and FPReg unit for the AMD-Athlon core.

IntReg estimator	FPReg estimator
$\text{TAAF}_{\text{est}}^{\text{IntReg}}(n) = 0.1976 * \text{IRA} + 0.77 * \text{TAAF}(n-1) + 0.947$	$\text{TAAF}_{\text{est}}^{\text{FPReg}}(n) = 0.636 * \text{NFP} + 0.832 * \text{TAAF}(n-1) + 0.627$

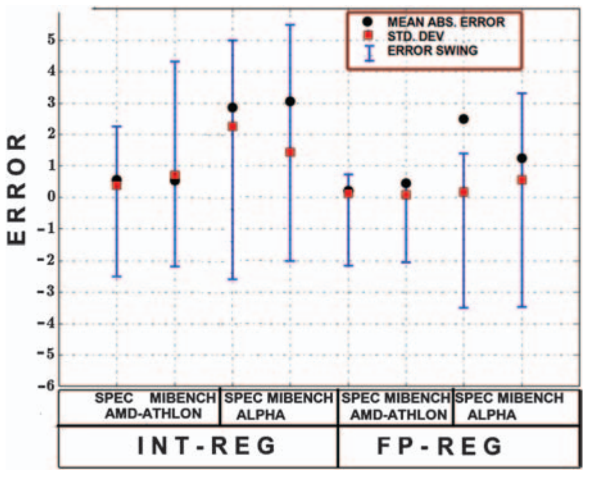

Fig. 14. Overall error variation in direct estimation of TAAF across alpha and AMD-Athlon processors for the IntReg and FPReg units.

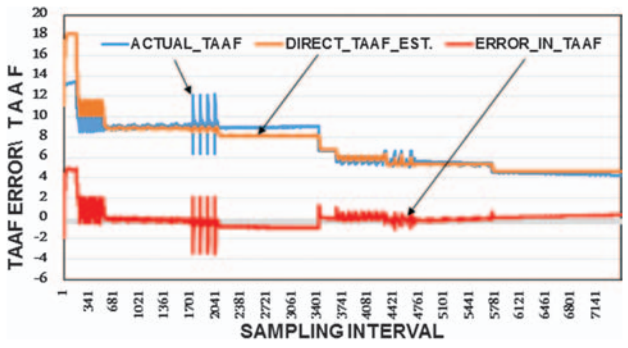

Fig. 15. Variation in actual TAAF, directly estimated TAAF and its estimation error over time for one of the testing set in the IntReg unit of AMD-Athlon processor.

7. CONCLUSION

In this paper we have presented an effective online temperature estimation technique using temperature history and performance counters. Our results show that the thermal history has a significant impact on reducing the offset term of the thermal estimator, preventing over/under estimation. We have also presented a dual estimator scheme that enhances the temperature estimation accuracy. Our technique is lightweight, requiring two or three multiplications and two additions.

In addition, we have discussed an approach to combine performance counters monitoring with on-chip sensor(s) to tolerate variability in ambient temperature. When properly selected and sampled at the appropriate rate, performance counters can form the basis of an efficient mechanism for accurately guiding thermal and reliability control algorithms in processors.

Acknowledgment: This work was partially supported by the National Science Foundation under grant CNS-1329831.

References

1. I. Koren and C. Krishna, Temperature-aware computing. *Sustainable Computing: Informatics and Systems* 1, 46 (**2011**).
2. M. Berktold and T. Tian, Cpu monitoring with dts/peci (**2010**), https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/cpu-monitoring-dts-peci-paper.pdf.
3. S. W. Chung and K. Skadron, Using on-chip event counters for high-resolution, real-time temperature measurements. *Thermal and Thermomechanical Phenomena in Electronics Systems 2006*, 114 (**2006**).
4. K. Singh, M. Bhadauria, and S. A. McKee, Prediction-based power estimation and scheduling for cmps, *Proceedings of the 23rd International Conference on Supercomputing*, ACM (**2009**), pp. 501–502.
5. C. Isci and M. Martonosi, Runtime power monitoring in high-end processors: Methodology and empirical data, *Proc. of the 36th Annual IEEE/ACM Int'l Symp. on Microarchitecture*, MICRO 36, IEEE Computer Society, Washington, DC, USA (**2003**), pp. 93–104.
6. R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, A study on the use of performance counters to estimate power in microprocessors. *IEEE Transactions on Circuits and Systems II: Express Briefs* 60, 882 (**2013**).
7. S. Yang, R. A. Shafik, S. Khursheed, D. Flynn, G. V. Merrett, and B. M. Al-Hashimi, Application-specific memory protection policies for energy-efficient reliable design, *2015 International Symposium on Rapid System Prototyping (RSP)*, October (**2015**), pp. 18–24.
8. W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, Hotspot: A compact thermal modeling method for cmos vlsi systems. *IEEE Transactions on* 14, 501 (**2006**).
9. H. Zhou, X. Li, C.-Y. Cher, E. Kursun, H. Qian, and S.-C. Yao, An information-theoretic framework for optimal temperature sensor allocation and full-chip thermal monitoring, *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June (**2012**), pp. 642–647.
10. JEDEC Solid State Technology Association Notice, 9 (**2007**), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.138.7840.
11. A. Das, B. M. Al-Hashimi, and G. V. Merrett, Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems. *ACM Trans. Embed. Comput. Syst.* 15, 24 (**2016**).
12. A. K. Coskun, T. S. Rosing, and K. C. Gross, Proactive temperature management in mpsocs, *Proceedings of the 13th International Symposium on Low Power Electronics and Design*, ACM (**2008**), pp. 165–170.
13. K.-J. Lee and K. Skadron, Using performance counters for runtime temperature sensing in high-performance processors, *19th IEEE International Parallel and Distributed Processing Symposium, 2005, Proceedings*, IEEE (**2005**), p. 8.
14. M. Bao, A. Andrei, P. Eles, and Z. Peng, On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature dependency consideration, *DAC '09, 46th ACM/IEEE Design Automation Conference, 2009*, July (**2009**), pp. 490–495.
15. A. Merkel, F. Bellosa, and Frank, Task activity vectors: A new metric for temperature-aware-scheduling, *Proc. of the 3rd ACM SIGOPS/EuroSys European Conf. on Computer Systems, Eurosys '08*, NY, USA (**2008**), pp. 1–12.
16. Y. Han, I. Koren, and C. M. Krishna, TILTS: A fast architectural-level transient thermal simulation method. *J. Low Power Electron.* 3, 13 (**2007**).
17. J. Lee, K. Skadron, and S. W. Chung, Predictive temperature-aware dvfs. *IEEE Tran. on Computers* 59, 127 (**2010**).
18. D. Upton and K. Hazelwood, Evaluating linear regression for temperature modeling at the core level, *In Workshop on Duplication, Deconstructing, and Debunking* (**2011**), pp. 8–14.
19. C. Zhuo, D. Sylvester, and D. Blaauw, Process variation and temperature-aware reliability management, *Design, Automation Test in Europe Conf. Exhibition (DATE), 2010*, March (**2010**), pp. 580–585.
20. M. A. Hall, Correlation-based feature subset selection for machine learning. Ph.D. Thesis, University of Waikato, Hamilton, New Zealand (**1998**).
21. F. J. Mesa-Martinez, E. K. Ardestani, and J. Renau, Characterizing processor thermal behavior, *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, ACM, ASPLOS XV, New York, NY, USA (**2010**), pp. 193–204.
22. J. D. Rodriguez, A. Perez, and J. A. Lozano, Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 569 (**2010**).
23. N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 1 (**2011**).
24. F. J. Mesa-Martinez, J. Nayfach-Battilana, and J. Renau, Power model validation through thermal measurements, *Proc. of the 34th Annual Int'l Symp. on Computer Architecture, ISCA '07*, ACM, New York, NY, USA (**2007**), pp. 302–311.
25. 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, *WWC-4, 2001 IEEE Int'l Workshop on Workload Characterization, 2001* December (**2001**), pp. 3–14.

26. Spec cpu2006 suite: The standard performance evaluation corporation, https://www.spec.org/cpu2006 (**2006**).

27. S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures, *42nd Annual IEEE/ACM Int'l Symp. on Microarchitecture, 2009. MICRO-42*, December (**2009**), pp. 469–480.

28. D. Zoni, S. Corbetta, and W. Fornaciari, Hands: Heterogeneous architectures and networks-on-chip design and simulation, *Proc. of the 2012 ACM/IEEE Int'l Symp. on Low Power Electronics and Design, ISLPED '12*, ACM, New York, NY, USA (**2012**), pp. 261–266.

29. G. Holmes, A. Donkin, and I. H. Witten, Weka: A machine learning workbench, *Info. Systems. Proc. of the 1994 2nd Australian and New Zealand Conf. on Intelligent*, November (**1994**), pp. 357–361.

30. H. Akaike, A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19, 716 (**1974**).

31. Y. Zhu and D. Albonesi, Localized microarchitecture-level voltage management, *ISCAS 2006, Proc. 2006 IEEE Int'l Symp. on Circuits and Systems, 2006*, May (**2006**), pp. 40–44.

32. J. Suykens and J. Vandewalle, Training multilayer perceptron classifiers based on a modified support vector method. *IEEE Transactions on Neural Networks* 10, 907 (**1999**).

33. S. Sharifi and T. S. Rosing, An analytical model for the upper bound on temperature differences on a chip, *Proc. of the 18th ACM Great Lakes Symp. on VLSI, GLSVLSI '08*, ACM, NY, USA (**2008**), pp. 417–422.

34. B. Datta and W. Burleson, Calibration of on-chip thermal sensors using process monitoring circuits, *Proceedings of the 11th International Symposium on Quality Electronic Design (ISQED) 2010*, March (**2010**), pp. 461–467.

35. Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, Interconnect lifetime prediction under dynamic stress for reliability-aware design, *Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design, ICCAD '04*, IEEE, Computer Society, Washington, DC, USA (**2004**), pp. 327–334.

Mayank Chhablani

Mayank Chhablani *received his B.S. degree in Electronics and Communication Engineering from University of Rajasthan, India, in 2009 and his M.S. degree in Electrical and Computer Engineering from University Of Massachusetts, Amherst, USA, in 2016. He has previously worked as a senior systems engineer with Infosys Ltd., India and is currently working as a senior design engineer at AMD, Inc., Boxborough, USA. His interests include computer architecture, thermal- and reliability-aware computing and machine learning.*

Israel Koren

Israel Koren *is currently a Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst, and an IEEE Fellow. He has been a consultant to numerous companies including IBM, Analog Devices, Intel, AMD and National Semiconductors. His research interests include Fault-Tolerant systems, Computer Architecture, VLSI yield and reliability, Secure Cryptographic systems, and Computer Arithmetic. He publishes extensively and has over 250 publications in refereed journals and conferences. He is an Associate Editor of the VLSI Design Journal, and Sustainable Computing: Informatics and Systems. He served as General Chair, Program Chair and Program Committee member for numerous conferences. He is the author of the textbook "Computer Arithmetic Algorithms," 2nd Edition, A.K. Peters, 2002 and a co-author of "Fault Tolerant Systems," Morgan-Kaufman, 2007.*

C. M. Krishna

C. M. Krishna *received his PhD from the University of Michigan in Electrical Engineering. He is on the faculty of the Electrical and Computer Engineering department of the University of Massachusetts. His technical interests include real-time and cyber-physical systems, performance evaluation, distributed systems, and data analysis.*