Power Agnostic Technique for Efficient Temperature Estimation of Multicore Embedded Systems

Abstract*—Temperature plays an increasingly important role in the overall performance of a computing system and in its reliability. Increased availability of multi- and many-core systems provides an opportunity to manage the overall temperature profile of the system by cleverly designing the application-to-core mapping and the associated scheduling policies. There are clear penalties associated with an uncontrolled temperature profile: a core reaching a critical temperature usually activates built in shut down or voltage and/or frequency scaling mechanisms to cool it down, thereby leading to unplanned performance loss of the system. Similarly, deep thermal cycles with high frequency lead to severe deterioration in the overall reliability of the system. Design space exploration tools are often used to optimize binding and scheduling choices based on a given set of constraints and objectives. These exploration tools rely on fast and accurate temperature estimation techniques. We argue that the currently available techniques are not an ideal fit to design space exploration tools, and suggest a system level technique which is based on application fingerprinting. It does not need any information about the processor floorplan, the physical and thermal structure, or about power consumption. Instead, its temperature estimation is based on a set of application-specific calibration runs and associated temperature measurements using available built-in sensors. Using extensive experimental studies, we show that our technique can estimate temperature on all cores of a system to within 5^oC, and is three orders of magnitude faster than state of the art numerical simulators like *Hotspot.

I. Introduction

Temperature has become a first order concern in an optimized use of modern microprocessors due to significantly higher power densities, see [1], [2]. Managing the temperature profile of a system is critical, specially in high performance safety critical applications, like embedded electronic control units (ECUs) in a modern automobile. These ECUs are generally mounted in areas where ambient temperature is already high, such as in the vicinity of the engine itself. This creates a situation where thermal affects of applications running on the ECU must be carefully controlled to achieve reliable and consistent computational performance, see [3]. Reliability considerations are now driving these systems towards multicore-based architectures, see [4], [5]. Thus, in general, a set of applications running on embedded multiprocessor architectures must be carefully analyzed for their impact on the temperature profile.

Different applications stress a given multiprocessor core differently, thereby giving rise to different temperature profiles over time. It may be even possible that excessively high temperatures lead to the phenomenon of "thermal runaway" causing physical destruction of the computing hardware, see [6]. Normally, a core experiencing temperatures near some critical value, automatically triggers dynamic temperature control techniques like shut-down or DFVS, see [7], leading to an unplanned loss of quality-of-service, see [8], [9]. Such performance disruptions make it hard to provide end-to-end performance guarantees about the system. Automatic triggering of these techniques can be avoided if one can ensure that the set of applications mapped to the cores of a multiprocessor, along with the scheduling algorithms used on each core, will never lead to a temperature increase beyond the critical values. Even if dynamic power reduction techniques are applied, the availability of a proper thermal analysis methodology will allow for a combined temperature and performance analysis that can be used to explore alternative mapping, scheduling and thermal management mechanisms.

Beyond the analysis and control of the maximal temperature, it is also important to investigate thermal cycles: the magnitude and frequency of temperature changes on a multiprocessor influences its reliability, see [10], [11], [12].

Embedded multiprocessor embedded systems with resource constraints generally do not have spare computational power available to decide online on task mapping and scheduling such that all temperature and performance metrics are met. Thus, a potentially large design space needs to be explored offline using temperature-aware exploration tools, They evaluate various mappings and select the one which is most suitable according to provided optimization criteria.

The overall objective of this work, therefore, is to develop a fast and yet accurate temperature estimation framework, which can be used in design space exploration iterations. The problem of correctly estimating the temperature profile of all cores in a given multiprocessor has traditionally been solved by using two common approaches. One solution is to use a low-level thermal simulator, like *Hotspot*, see [13]. Building a numerical temperature simulator for a given multiprocessor requires detailed knowledge of the floorplan and electrical characteristics such as technology node, rail voltage, materials used, and power consumption of each micro-architectural unit in the processor, just to name a few. This information is not easily available, requiring designers to approximate the physical and electrical characteristics of the processor, which may lead to unacceptable inaccuracies of the estimated temperatures. Numerical simulators also tend to be too slow to be used in an iterative design-space exploration tool.

It is generally feasible to get total power consumption of the processor by using any of the measuring techniques described in [14], [15]. On the other hand, without detailed circuit and hardware implementation information, it is not possible to get an accurate breakdown of the total power amongst the micro-architectural units (power density distribution). Consequently, abstract temperature models have been reported in [16] which attempt to estimate the temperature profile as a function of total power consumption of each core. This kind of abstract models can be computed quickly, but leads to a high degree of inaccuracy, since it considers a few observable parameters only to calculate the temperature, e.g. the total power consumption of a processor. We argue that it is not possible to build a sufficiently accurate thermal model by making use of this kind of coarse-grained abstraction.

A. Motivational Example

The approach used in this work is unique, since it does not abstract away power density distribution information as a given application executes on a core of a multiprocessor. Such abstraction can lead to a high degree of inaccuracy as shown in the following sections. To motivate the discussion, a commonly used hardware architecture is used, which allows us to clearly identify deficiencies of current approaches. Please note that the new techniques that will be described in later sections do not need information about hardware details like floorplan or power-density information.

1) Correct Temperature Trace Estimation: Let us consider a four-core chip-multiprocessor (CMP) on which three applications are running denoted as *producer*, *FFT* and *consumer*, see Figure I-A1. The *producer* is in charge of creating data for the *FFT* application, which in turn supplies the results to the *consumer* application for display. Both, *producer* and *consumer*, are I/O intensive applications and it can be expected that a considerable amount of power is consumed in data caches. On the other hand, *FFT* is a compute-intensive application and the ALU will dominate the power consumption of the corresponding processor core. All three applications consume the same total amount of power.

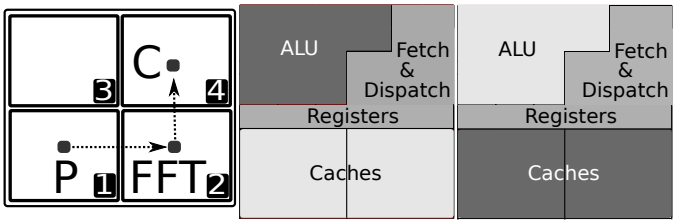

Figure 1. Producer (P), FFT and Consumer (C) applications run on a CMP. The power density distribution in micro-architectural components of the cores 1 and 2 is shown, where lighter shades show areas with higher power consumption.

The left floorplan shows the power consumption on core 1 when *producer* is running, while the right floorplan shows the power distribution on core 2 when FFT executes. The temperature sensor is located near the upper left corner of the processor. hence, it is more sensitive to the heat generated by the computational units of the processor.

According to frequently used coarse grained temperature estimation techniques such as those in [17], [16], the temperature is estimated as a function of total power consumption of a core i according to

$$T_i(t) = f_i\left(\mathcal{P}_1(t), \ldots, \mathcal{P}_n(t)\right) \quad (1)$$

where $\mathcal{P}_i(t)$ denotes the total power consumption of processor core i in the CMP at time t.

The temperature trace on the cores 1, 2 and 4 is shown in Figure I-A1. The temperature simulation was performed on Hotspot using the Alpha 21264 physical and thermal structure and power model as supplied with Hotspot. Power numbers were generated from the Wattch/Simplescalar tool chain [18]. Temperature influence from cores to their neighbors has been accounted for. All applications run according the same schedule: tasks execute in lock-step, i.e. cores 1, 2, and 4 have the same total power trace over time. Communication between cores is implemented using FIFO buffers. In such a scenario, a simple coarse grained model such as in (1) can not be expected to yield accurate results since it is oblivious to the power density distribution in the cores. The model will predict the same temperature trace for cores 1, 2 and 4, which may be similar to *one* of the temperature traces shown in Figure I-A1 or an average trace, depending on how f_i is constructed. The temperature estimates due to (1) will be inaccurate even if all the applications in the example were executed on the same core.

We can conclude that ignoring the power density distribution between the micro-architectural units can lead to large errors in the temperature estimates. On the other hand, detailed power distribution of total power in a processor core is hardly available for any modern processor as it would necessitate detailed circuit and physical models.

2) Platform Constraints: A system designer may be interested in evaluating the temperature trace due to *producer, consumer* and *FFT* on newer embedded CMPs, such as Nvidia-Tegra used in mobile devices, or the multicore ARM (e.g., Cortex A9) processors. Without the knowledge of detailed floorplan information, detailed power information (at the granularity of micro-architectural units) as well as detailed thermal models, numerical simulators cannot be used with sufficient accuracy. As shown above, simple abstract models such as (1) exhibit limited accuracy.

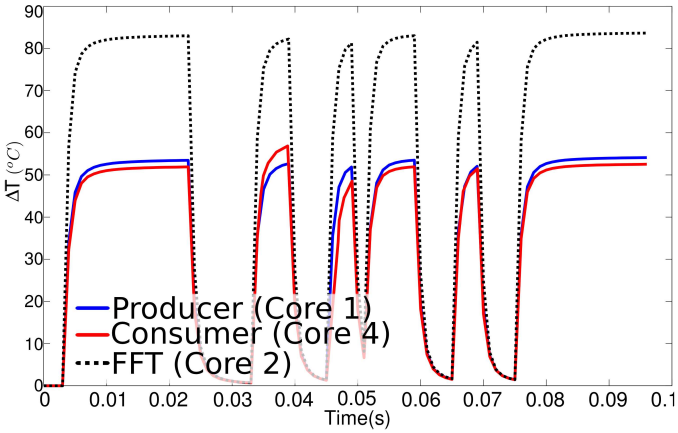

Figure 2. Temperature trace on four cores due to running applications *Producer, Consumer* and *FFT*.

3) Computation Time: The temperature trace in the previous example was calculated using Hotspot at a temporal resolution of $1ms$, which took approximately 6 hours to complete even in case of a relatively short trace of $100ms$. Design space exploration is usually an iterative process in which many sample candidate mappings are investigated. Therefore, this class of numerical simulators is not appropriate due to their long run-time. In order to be useful for design space exploration, a temperature estimation method should represent a different trade-off in terms of speed and accuracy. In addition, it should not rely on often unavailable information such as detailed power, layout, physical and thermal models.

B. The Problem

Based on the above discussion, the problem that needs to be solved can be described as follows:

> Given a possibly heterogeneous chip multiprocessor system $\mathcal{S}$ and a set of applications $\mathcal{A}$: estimate the temperature trace on all cores of $\mathcal{S}$ with sufficient speed and accuracy as required for design space exploration. The method should not depend on prior availability of power, layout, physical and thermal models of the hardware platform.

As we will see below, the fingerprinting approach as proposed in this paper replaces the detailed knowledge about platform internals by a limited set of calibration runs where applications are executed on the platform and temperature traces from the internal sensors are recorded.

C. Our Contribution

An iterative design space exploration (DSE) may not use numerical simulators due to their high computational requirement. Therefore, abstract solvers which calculate temperature traces based on limited information are the preferred candidates for integration into DSE tools. Thus, the focus of this work is to develop a method to determine the thermal system behavior with sufficient speed and accuracy such that many mappings of $\mathcal{A}$ onto $\mathcal{S}$ can be quickly evaluated. In particular, we demonstrate a technique to build a thermal model

relying on the results of a limited set of calibration runs including temperature measurements. This model is combined with mapping and scheduling information and results in the desired estimated temperature traces.

Specifically, our contribution is a temperature evaluation framework which:

- Can correctly determine correct a temperature trace even when two applications running on CMP consume the same total power but exercise different micro-architectural units;
- Does not require knowledge of power traces, and does not assume that all cores of the CMP are homogeneous;
- Can model and evaluate temperature effects of various mapping policies in terms of peak temperature, dynamic temperature range both in space and time;
- Does not depend on prior knowledge of details about the hardware platform;
- Allows for fast and accurate temperature estimation to be reliably used in DSE loops.

II. Related Work

Temperature estimation has been recent focus of research, due to the reasons discussed above. Overall, the estimation techniques are based on numerical simulation, or based on abstract relationships between power and temperature such as (1).

Numerical simulators model the entire multiprocessor as a complex resistor-capacitor network (eg, Hotspot) and calculate temperature by numerically solving a large set of differential equations. These simulators depend on knowledge of the exact power consumption of each micro-architectural unit within each core. These power numbers may be obtained for certain processors using the Wattch/Simplescalar toolchain. For other processor designs, a generally accepted method has been to use hardware sniffers, which calculate the number of times a micro-architectural unit has been accessed by an application, see [19]. However, this requires setting up special registers using software or hardware methods that record the accesses of all micro-architectural entities in the processor, which may not be always possible. In addition, unless dedicated hardware is used, the application of measuring the access count may disturb the behavior of the profiled application itself, thus, a model derived out of such a scheme may not be accurate. The combination of large computational power required for numerical simulators, in addition to detailed knowledge of the hardware as well as software make this approach unfeasible for design space exploration. A System-C based thermal simulator has recently been reported, but suffers from the same basic limitation as other simulators: the level of detailed information required for setting up the model is not easily available, see [20].

Computationally fast simulators based on first-order differential equations have been used in [16], but their applicability to modern CMP systems is not clear since the thermal model is too simplistic to take all important temperature dynamics into account, like differences in utilization of core's micro-architectural units. Li et al. propose an abstraction based approach, which builds a thermal model based on total power consumption of the processor, which is used to calculate temperature traces for given applications, see [21]. However, Li's approach does not distinguish between the differences in the spatial power profile of applications, i.e. two applications consuming the same total power but targeting different micro-architectural units are indistinguishable. Such an abstraction can lead to large errors in the estimated temperature of cores, as already discussed in Section I-A.

A look-up table based approach involves building resistor-capacitor (RC) models for $\mathcal{S}$, and recording tables of time-temperature relationship for the set of applications, $\mathcal{A}$, see [22]. The resulting estimation methods are fast, but again, they assume that a unique total power consumption always implies a unique temperature distribution. Separate databases are created for temperature increments (when application raises the temperature of the CMP) and for temperature decrements (application is not active). Since temperature changes depend on the current temperature of a core, it is not clear as how to how much data is required to model all possible switching scenarios.

Thus, conventionally available solutions either assume the availability of hard to get information (numerical simulators based approach) or are too abstract for estimating correct temperature traces (abstract power-model based models).

III. Setup and Notations

In the following sections, we assume an arbitrary chip multiprocessor $\mathcal{S}$ consisting of $\mathcal{N} \in \mathbb{N}$ cores $\mathcal{S}_j \in \mathcal{S}$. It is not necessary that all cores in $\mathcal{S}$ are homogeneous. A core, for instance may be of type graphics processor (GPU), a floating point processor (FPU), a RISC processor etc. Thus, we have available a set of processor types, $\mathcal{C} = \{GPU, FPU, RISC, ...\}$ available on $\mathcal{S}$. A function $\mathcal{T} : \mathcal{S} \rightarrow \mathcal{C}$ maps the set of cores in $\mathcal{S}$ to their types. Also available is the set of applications $\mathcal{A}$ that may execute on $\mathcal{S}$. The i^{th} application in $\mathcal{A}$ is referred to as $\mathcal{A}_i$. The approach taken in the work treats each application $\mathcal{A}_i \in \mathcal{A}$ as a black-box to be run on $\mathcal{S}$. Thus, this approach can be used to estimate temperature for an arbitrary given set of applications. All applications bound to a given core may be scheduled according to a scheduling policy such as *earliest deadline first* (EDF), *round-robin* (RR), *least laxity first* (LFF) or *rate-monotonic* (RM).

$P(\mathcal{A}_i, \mathcal{S}_j)$ denotes the instantaneous total power consumption of core $\mathcal{S}_j$ due to an application $\mathcal{A}_i$. $\mathcal{P}(\mathcal{A}_i, \mathcal{S}_j)$ refers to time trace of instantaneous total power consumption of core $\mathcal{S}_j$ due to the application $\mathcal{A}_i$; henceforth referred to as 'power trace'. We suppose that an application consumes constant power as long as it is running. The utilization alphabet, $\mathcal{U} \in \{0, 1\}$ represents the utilization of a core by an application for a time interval with length t_s. In other words, if an application is running then its utilization is 1, otherwise 0. The time-trace of an application $\mathcal{A}_i$ is given by $\sum_{\mathcal{U}} \mathcal{A}_i$, which is a tuple whose elements are in $\{0, 1\}$. The set of all tuples is denoted by $\sum_{\mathcal{U}}^{*}$. In other words, $\sum_{\mathcal{U}} \mathcal{A}_i$ is the time-trace of the utilization of application $\mathcal{A}_i$, specified with a given time-resolution t_s.

If $\mathcal{S}$ has a square or a rectangular physical footprint, the location of a core in $\mathcal{S}$ can also be specified in Cartesian co-ordinates <x,y>, with the origin located at the lower left corner of $\mathcal{S}$. In this case, two cores with co-ordinates <x,y> and <x',y'>, respectively, are said to be k hops apart, if $k = \max\{|x' - x|, |y' - y|\}$.

A set of temperature sensors $\mathcal{R}$ is available on $\mathcal{S}$. Temperature for core $\mathcal{S}_j$ is available from $\mathcal{R}_j$. It is assumed that reading from $\mathcal{R}_j$ represents the temperature for that core. Logging of temperature trace is done only during the construction of the thermal model.

We also define the severity of thermal cycles experienced by the chip multiprocessor. Thermal cycles are periodic changes in temperature experienced by a core $\mathcal{S}_j$, when a given subset $\mathcal{A}' \subseteq \mathcal{A}$ of applications execute on it. Large variations in temperature are said to be worse for hardware reliability, as compared to small ones. The hardware is designed to withstand a certain maximum number of such temperature cycles, before it fails, see [23]. Therefore, a core $\mathcal{S}_j$ has a fixed "thermal-cycle budget", and the entire system's budget is simply the sum total of the thermal-cycle budgets for each core. Based on this concept, a simple metric that measures the "expenditure" from the total thermal-cycle budget, is $\mathcal{V}$:

$$\mathcal{V} = \sum_{i=1}^{\mathcal{N}} (\Delta T_i) \times f_i \tag{2}$$

ΔT_i is the maximum temperature variation experienced by a core

$\mathcal{S}_i$; f_i is the frequency of this temperature variation and '$\times$' is the multiplication operator. A mapping with smaller $\mathcal{V}$ is preferable.

IV. Application Fingerprinting

This section describes the construction of the thermal model of chip multiprocessor $\mathcal{S}$, given a set of applications, $\mathcal{A}$. We call this technique "application fingerprinting". Application fingerprinting assumes that the thermal model of $\mathcal{S}$ is linear. It has been shown that a thermal model of a processor can be constructed by using only passive electrical components, such as a resistor and a capacitor, see [24]. It is also known that any mesh consisting of such passive electrical components forms a linear circuit, thereby justifying our assumption, see [13], [25].

The overall idea of application fingerprinting is to determine the thermal impulse response, $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) \forall\, i, p, q$ such that the temperature trace $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$ due to $\mathcal{A}_i$ can then be calculated easily:

$$T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) = \quad \sum_{\mathcal{U}} \mathcal{A}_i \quad \otimes \quad H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) \tag{3}$$

where $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$ is the required thermal impulse response for the application $\mathcal{A}_i$, when it executes on core $\mathcal{S}_p$ and the resulting temperature change is calculated for core $\mathcal{S}_q$. The symbol $\otimes$ is the convolution operator. Notice that power trace is not used in the calculation of the temperature. The utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$, the impulse response, and the calculated temperature trace are all given at the time resolution of t_s.

Presented below are two claims that enable the calculation of accurate temperature traces, without requiring any knowledge of the power density distribution in a core, or its power trace.

These claims are justified in the following sections. Data extracted from the Simplescalar/Wattch simulator is used to support the claims being made on the the relationships between total power, temperature and power-densities, due to an application such as $\mathcal{A}_i$.

A. Non-Unique Relationship between Total Power Trace and Temperature Trace

A power trace associated with an application $\mathcal{A}_i$, executing on a core $\mathcal{S}_j$, does not automatically imply a unique temperature trace, on any core in the chip multiprocessor $\mathcal{S}$. The power density distribution in the core determines the net flow of heat between various parts of the core, and hence, the overall temperature trace. Multiple applications can have the same total power consumption, but different power density distributions, causing a different overall temperature trace. An example was already discussed in section I-A. Thus, a correct thermal model must not calculate temperature traces solely as a function of various power traces when the set of applications $\mathcal{A}$ execute on the system $\mathcal{S}$.

In summary, given a temperature trace $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$, the application $\mathcal{A}_i$ may not be unique. However, the following relationship is deterministic: given $\mathcal{A}_i$ executes on core $\mathcal{S}_p$, its power trace is always $\mathcal{P}(\mathcal{A}_i, \mathcal{S}_p)$.

B. Unique Relationship between application and Relative Power Distribution

An application $\mathcal{A}_i$ executing on core $\mathcal{S}_p$, consumes a constant total instantaneous power, $P(A_i, \mathcal{S}_p)$. Furthermore, a given $P(A_i, \mathcal{S}_p)$ uniquely determines the relative distribution of this total power amongst the core's micro-architectural units. We suppose that a particular application, such as a 16-point FFT, will run through the same sequence of steps, irrespective of the inputs. We show that such an assumption is a reasonable abstraction. In case the input to this application varies, these sequence of steps are repeated appropriately. Such repetition of the sequence of steps also causes the number of accesses to each of $\mathcal{S}_p$'s micro-architectural units to scale appropriately.

S.No	Application	#Runs	ΔP_{max}	Instructions Executed
1	FFT	10	4%	$64,892$ $354,675$
2	I-JPEG	10	2.7%	6.2×10^6 119.6×10^6
3	Matrix-Multiplication	10	1.2%	$85,910$ 107.6×10^6
4	GSM-Encoder ("toast")	10	0.8%	5.6×10^6 19.6×10^6

Table I
Power distribution statistics of selected benchmarks.

This claim was validated using several benchmarks from the MiBench Embedded Systems benchmarks suite, see [26]. The results for selected benchmarks are presented in Table I. These benchmarks were run with varying inputs, which is reflected in the number of instructions executed (column 5, minimum number of instructions vs maximum number of instructions executed). The variation in instantaneous total power consumption of an application executing under varying inputs is minimal. For instance, the maximum variation for the FFT application was only 4%, with inputs ranging from single-digit values to six-digit values. Similar results are obtained for other benchmarks.

In addition, power consumption per instruction for each of these benchmarks was also evaluated after making suitable modifications to the Wattch simulator. The results for FFT and GSM-Encoder ("toast") application are shown in Figure 3. The same conclusions apply for other applications. It can be seen from the figure that the mean power consumption in all micro-architectural units in the core remains almost constant even under significant input variations. Almost all the difference in any total power consumption can be attributed to the variation in power consumed by the clock. Statistical parameters such as mode, median and standard-deviation are also shown in Figure 3. It can be observed that these statistical parameters also remain relatively constant.

From the preceding discussion, the following conclusions can be drawn:

- The instantaneous total power consumption of $\mathcal{A}_i$ executing on core $\mathcal{S}_p$ is:

$$\begin{cases} P(\mathcal{A}_i, \mathcal{S}_p) & : \quad \mathcal{U} = 1 \\ 0 & : \quad \mathcal{U} = 0 \end{cases} \tag{4}$$

- For an application $\mathcal{A}_i \in \mathcal{A}$, executing on core $\mathcal{S}_p$, its utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$ also determines its power trace, one varying from the other only by a scalar. Specifically:

$$\mathcal{P}(\mathcal{A}_i, \mathcal{S}_p) \quad = \quad s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p)) \times \quad \sum_{\mathcal{U}} \mathcal{A}_i \tag{5}$$

where $s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p))$ is scalar depending on $\mathcal{A}_i$, and the type of processor core $\mathcal{T}(\mathcal{S}_p)$.

- Assume an thermal impulse response $H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$ determined from power-trace $\mathcal{P}(\mathcal{A}_i, \mathcal{S}_p)$ and temperature trace $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$. Also assume another impulse response $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$, determined from $\sum_{\mathcal{U}} \mathcal{A}_i$ and $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$. Requiring that the temperature traces calculated using either impulse responses must be equal:

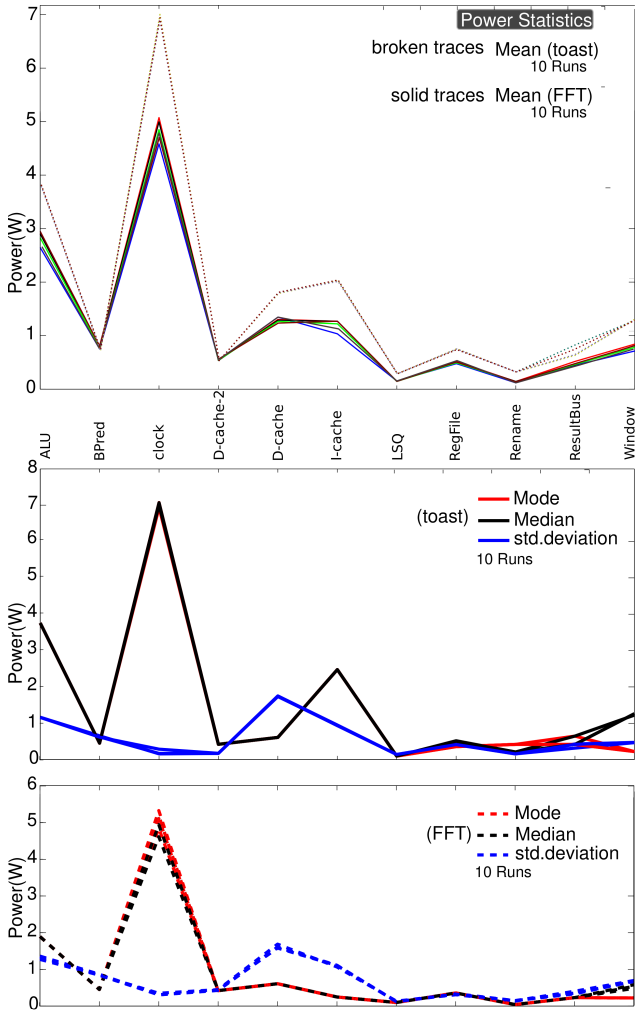

Figure 3. Statistics on power consumption for major micro-architectural units. Mean power consumption for 10 benchmarks (top), and main statistical parameters power consumption for the same runs (bottom).

$$\sum_{\mathcal{U}} \mathcal{A}_i \otimes H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) = \mathcal{P}(\mathcal{A}_i, \mathcal{S}_p) \otimes H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$$
$$= \{s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p)) \times \sum_{\mathcal{U}} \mathcal{A}_i\} \otimes H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$$
$$= \sum_{\mathcal{U}} \mathcal{A}_i \otimes \{s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p)) \times H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)\} \quad (6)$$

Therefore, we find:

$$H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) = s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p)) \times H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) \quad (7)$$

and:

$$T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) = \mathcal{P}(\mathcal{A}_i, \mathcal{S}_p) \otimes H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$$
$$= s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p)) \times \sum_{\mathcal{U}} \mathcal{A}_i \otimes H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q))$$
$$= \sum_{\mathcal{U}} \mathcal{A}_i \otimes \{s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p))) \times H'(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q))\} \quad (8)$$

In other words, (5), (7) and (8) taken together show that the impulse response $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q))$ determined using the temperature trace, $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q))$ and the utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$ can be used to estimate correct temperature traces, without requiring any power trace information.

C. Estimating Impulse Responses

Figure 4 shows the overview of the application fingerprinting technique. The technique starts with taking the set of applications, $\mathcal{A}$ and the system $\mathcal{S}$, for estimation of impulse responses, $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) \forall i, p, q$. Each application $\mathcal{A}_i \in \mathcal{A}$ is run individually on $\mathcal{S}_p$ using a known utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$. As the application $\mathcal{A}_i$ executes, temperature traces from all other cores are $\mathcal{S}$ recorded. The estimation of impulse responses is based on the *Generalized Pencil-of-functions* (GPOF) technique, see [27]. Our estimation approach is summarized in Algorithm 1. All impulse responses are collected in a three dimensional matrix, H.

Notice that the procedure for estimation of impulse responses is different from Li's approach, see [21]. Li requires power trace information, whereas we require only utilization traces and corresponding temperature traces from $\mathcal{S}$. Impulse responses are estimated for each application $\mathcal{A}_i \in \mathcal{A}$, making it possible to account for different power density distributions, even when there are multiple applications consuming the same total power. As a consequence, it is possible to avoid incorrect calculations of temperature traces, as discussed in section I-A. Note that since impulse responses are calculated using the utilization trace of an application and the corresponding temperature trace, the scalar, $s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p))$ is automatically accounted for.

Once all impulse responses are available, the correct temperature trace can be calculated for any candidate mapping consisting of applications from $\mathcal{A}$, executing on $\mathcal{S}$. Subsequent temperature trace calculations require the knowledge of only the utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$ of the core on which the application $\mathcal{A}_i$ executes. Note that power traces are not required for the temperature estimation step.

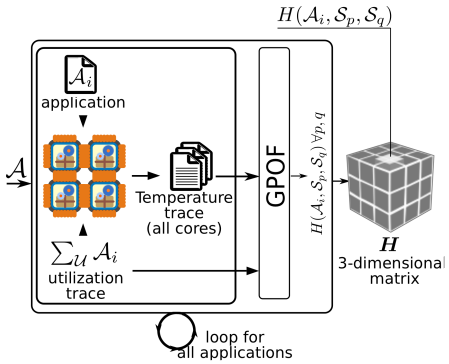

Figure 4. A set of impulse response matrices for application set $\mathcal{A}$ is created during the fingerprinting application.

V. Temperature Aware Design Space Exploration

With all required impulse responses available, design space exploration can be performed to evaluate the effect of various mappings on the temperature profile over the chip multiprocessor, $\mathcal{S}$.

DSE tools are already available, which, given a set of applications, and a set of abstract hardware properties (number of cores, type of cores etc, but not the detailed floorplan) calculate various mappings subject to a set of constraints and objectives, see [28]. However, such tools usually are not temperature aware.

A temperature aware DSE loop is shown in Figure 5. The DSE tool accepts the following parameters:

1) Abstract architectural properties: available computing resources, their types etc;
2) Set of mapping constraints and objectives;

Algorithmus 1 Algorithm to determine $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$

1: **begin**
2: $\mathcal{A}$: set of applications
3: $|\mathcal{A}|$: cardinality of $\mathcal{A}$.
4: $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$: temperature trace of application $\mathcal{A}_i$ running on core $\mathcal{S}_p$, with temperature measured on core $\mathcal{S}_q$
5: $\sum_{\mathcal{U}} \mathcal{A}_i$: utilization trace of application $\mathcal{A}_i$ running on core $\mathcal{S}_p$.
6: **do** for $i = 1 : |\mathcal{A}|$// *Iterate over all applications*
7: **do** for $p = 1 : \mathcal{N}$ *//Iterate for all cores in* $\mathcal{S}$
8: Run $\mathcal{A}_i$ on core $\mathcal{S}_p$
9: $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q) = GPOF(\sum_{\mathcal{U}} \mathcal{A}_i, T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)), \forall q$ *//*$s(\mathcal{A}_i, \mathcal{T}(\mathcal{S}_p))$ *is automatically accounted for*
10: **end**
11: **end**
12: `Procedure`: *GPOF*(PowerTrace, TemperatureTrace)
13: *// Calculates impulse response from utilization trace and temperature trace based on the generalized pencil-of-functions algorithm.*
14: *// returns the estimated impulse response.*
15: **end**

3) Set of applications, $\mathcal{A}$;
4) Evaluated temperature characteristics of a supplied mapping, $\mathcal{M}$. A mapping $\mathcal{M}$ provides the following information:
   - Binding for all applications in $\mathcal{A}$. That is, each application in $\mathcal{A}$ is provided with a core on which it will execute.
   - Scheduling policy for each *core*, such as EDF, RR, LFF etc.

A candidate mapping generated by the DSE tool is evaluated using the temperature evaluation component. The detailed algorithm for calculation of temperature traces is presented in the next section. Based on the feedback of the temperature evaluation component, the DSE tool may modify its internal parameters to rule out combinations that lead to unacceptable temperature profiles on $\mathcal{S}$. Or, the DSE tool may successively refine mappings that are deemed to be favorable in terms of temperature. A simple approach used in successive refinement of mappings is simulated annealing. In this approach, starting from an initial mapping $\mathcal{M}_0$, applications are moved between cores, and temperature traces are re-calculated. The DSE tool also evaluates different scheduling policies for each core. The process continues till the required performance objectives are met (viz., minimizing peak temperature, minimizing thermal cycles).

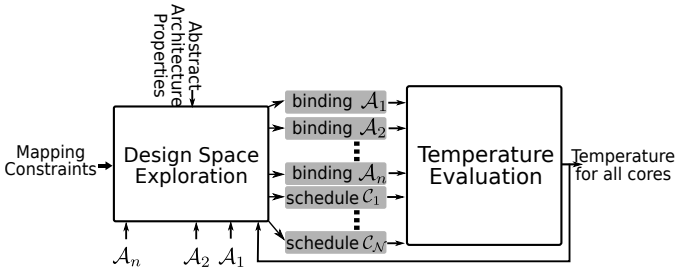

Figure 5. Temperature aware DSE Loop.

A. Temperature Trace Calculation from a given Mapping

The linearity property of the thermal model of $\mathcal{S}$ allows us to use the superposition principle for determining the overall temperature trace due a given mapping, $\mathcal{M}$. The procedure for calculating the detailed temperature trace is given in Algorithm 2.

The process of temperature trace calculation starts with a candidate mapping provided by the DSE. For a given core $\mathcal{S}_p$, its scheduling policy determines the utilization trace for each application bound to $\mathcal{S}_p$. The Cheddar project provides good libraries for automating the construction of such utilization traces, see [29].

Referring to Algorithm 2, lines 2-8 define the required variables. Line 11 initializes a loop to iterate over all applications to be run on

Algorithmus 2 Temperature estimation for $\mathcal{S}$, given a mapping $\mathcal{M}$

1: **begin**
2: $\mathcal{M}$: a mapping from DSE.
3: $\mathcal{M}_B(\mathcal{A}_i)$: Core to which $\mathcal{A}_i$ is bound
4: $|\mathcal{M}(\mathcal{A})|$: total number of applications bound to $\mathcal{S}$.
5: $U \epsilon (\sum_{\mathcal{U}})^{|\mathcal{M}(\mathcal{A})|}$: Array of utilization traces. Contains $|\mathcal{M}(\mathcal{A})|$ traces.
6: $U(\mathcal{A}_i) \epsilon \sum_{\mathcal{U}}^{*}$: Utilization trace for $\mathcal{A}_i$.
7: $\mathcal{S}_q \epsilon \mathcal{S}$: Core in $\mathcal{S}$ on which temperature is to be estimated.
8: T_q: Temperature trace on core $\mathcal{S}_q$.
9: **U = Cheddar($\mathcal{M}$)**
10: **init:** $T_q = 0 \, \forall q$
11: **do** for $i = 1 : |\mathcal{M}(\mathcal{A})|$ *//Iterate for all applications*
12: **do** for $q = 1 : \mathcal{N}$ *//Iterate for all cores*
13: $\overrightarrow{T}_q = \overrightarrow{T}_q + H(\mathcal{A}_i, \mathcal{M}_B(\mathcal{A}_i), \mathcal{S}_q) \otimes U(\mathcal{A}_i)$ *//Calculate temperature trace on* $\mathcal{S}_q$ *due to* $\mathcal{A}_i$ *on core* $\mathcal{M}_B(\mathcal{A}_i)$.
14: **end for**
15: **end for**
16:
17: **function Cheddar($\mathcal{M}$)**
18: *Return utilization trace for each application in* $\mathcal{M}$*, using libraries from Cheddar project.*
19: **end**

$\mathcal{S}$. Line 12 iterates over each core in $\mathcal{S}$, calculating the temperature trace due to $\mathcal{A}_i$, on all cores of $\mathcal{S}$ (Line 13). The algorithm loops till all applications have been accounted for, and the overall temperature trace on each core due to mapping $\mathcal{M}$ is calculated by superposition.

B. Sources of Inaccuracies

1) Inexact Impulse Responses: Estimation of the impulse response from a given utilization trace and an associated temperature trace measurement, is often an approximate process. Further, the order of the thermal model of $\mathcal{S}$ is limited to avoid dealing with overly complex impulse responses, thereby saving some computational effort. In this work, the accuracy of estimated impulse response, $H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)$ is specified as *Quality of Fit* (QoF):

$$QoF_{H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)} = 100 \left[1 - \frac{N}{D}\right] \quad [\%] \qquad (9)$$

Where:

$$N = ||T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_m - T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_e||$$

$$D = ||T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_m - \overline{T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_m}||$$

$T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_m$ is the measured temperature trace, due to a known utilization trace, $\sum_{\mathcal{U}} \mathcal{A}_i$. The mean value of the measured temperature trace is given by $\overline{T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_m}$. The temperature trace, $T(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q)_e$ is calculated using the estimated impulse response, and the utilization trace $\sum_{\mathcal{U}} \mathcal{A}_i$. A QoF of 100% indicates a perfect impulse response. The QoF depends on the p, q, $\mathcal{A}_i$ and on the order of the thermal model. It also depends on the utilization trace $\sum_{\mathcal{U}} \mathcal{A}_i$. The QoF reported in the experiments section is the worst QoF calculated over several utilization traces, ranging from $\sum_{\mathcal{U}} \mathcal{A}_i = [1, 1...1]$ (application is always executing) to $\sum_{\mathcal{U}} \mathcal{A}_i = [1, 0, 1, 0...]$ (start-stop execution of application at time interval of t_s).

2) Under-estimating the impact of a hot core on a distant neighbor: It can be shown analytically, as well as from results published in recent literature, that temperature on a core drops rapidly with distance from the temperature hotspot, see [30]. This is attributed to high lateral thermal resistance. If from section IV-C, it is determined that the lateral thermal resistance of $\mathcal{S}$ is very high, it becomes tempting to ignore the temperature effects of an active core on cores far away from itself. This reduces the computational effort for the computation of temperature traces, but at the cost of accuracy. In this case, the maximum possible error that can be incurred in temperature calculations must be determined. Assuming that we would like to

Algorithmus 3 Worst Case Error Estimate.

1: **begin**
2: k: hops beyond which temperature affect of an active core is ignored.
3: $\mathcal{S}_p$: A core in $\mathcal{S}$.
4: $O_{c,\,c'}$: Observer core with location $c,\ c'$, at the center of $\mathcal{S}$.
5: $\mathcal{A}$: set of applications. $\mathcal{A}_i$: i^{th} application in $\mathcal{A}$.
6: $|\mathcal{A}|$: cardinality of $\mathcal{A}$.
7: E^*: Maximum error in temperature estimation, observed at $O_{c,\,c'}$.
8: $\mathcal{H}^k = \{C_{x',\,y'} | \, (|y' - c'| = k) \, || (|x' - c| = k)\}$ //set of all cores k hops away.
9: L is the largest hop distance in the $\mathcal{S}$, w.r.t. $O_{c,\,c'}$.
10: // From all applications in $\mathcal{A}$, determine which one leads to highest temperature rise.
11: **do** for i = 1: $|\mathcal{A}|$
12: $T^*(i) = FV(H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_p))$
13: **end for**
14: $\mathcal{A}^*$: application that leads to highest $T^*(i)\ \forall i$.
15: **do** for i = k+1:L // go over all hop distances from $k+1$ outwards
16: **do** for j=1:$|\mathcal{H}^i|$ // go over cores at this hop distance
17: $E^* = E^* + FV(H(\mathcal{A}^*,\ \mathcal{H}^i_j,\ O_{c,\,c'}))$// $\mathcal{H}^i_j \, \epsilon \, \mathcal{H}^i$, is j^{th} core in set $\mathcal{H}^i$.
18: **end for**
19: **end for**
20:
21: `Procedure` $FV(H(\mathcal{A}_i, \mathcal{S}_p, \mathcal{S}_q))$
22: Return steady-state temperature on core q due to $\mathcal{A}_i$ running on $\mathcal{S}_p$.
23: //Calculated using final-value theorem. Steady-state temperature is the highest temperature any core will experience due to $\mathcal{A}_i$ running continuously.
24: **end**

ignore the temperature affects due to an active core beyond distances of k hops, the worst case error in temperature estimates due to such an assumption must be calculated as shown in Algorithm 3.

Referring to Algorithm 3, the worst case error is estimated at an 'observer core', $O_{c,c'}$, at the center of $\mathcal{S}$. Since the temperature affect of an active core reduces with the hop distance from itself, a centrally located core will have maximum 1-hop neighbors, maximum 2-hop neighbors and so on. Further, uniform cooling over $\mathcal{S}$ is assumed, which ensures that the worst case error in temperature estimate is not missed. Final Value theorem for transfer functions is used to determine the maximum possible temperature influence of an active core on its neighbors. Line 12 determines the steady-state temperature due to application $\mathcal{A}_i$, running on a core $\mathcal{S}_p$. Any core $\mathcal{S}_p$ within $\mathcal{S}$ may be chosen. Line 14 determines the application $\mathcal{A}^*$ which leads to the highest steady-state temperature on core $\mathcal{S}_p$. Line 17 then calculates the error by calculating the accumulated influence of all cores beyond k hops from $O_{c,c'}$. The algorithm assumes that all cores which lie more than k hops from $O_{c,\,c'}$ are running $\mathcal{A}^*$. The values of E^* with respect to hop distance are shown in Figure 6.

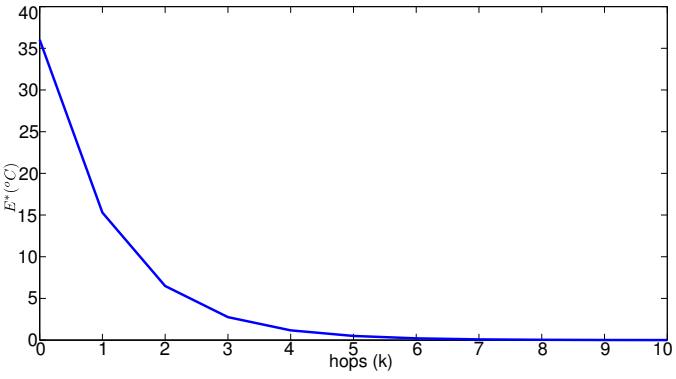

Figure 6. Maximum error in temperature estimate at $O_{c,\,c'}$ with hop-distance beyond which it is assumed that an active core produces no temperature affect.

The values in Figure 6 are specific to our experimental platform, but the nature of the curve is expected to remain the same for any chip-multiprocessor platform. The results clearly show the risk associated with making any uncalculated simplifications on the impact of a hot core on its neighbors.

Application	0-hop	1-hop	2-hop	3-hop	> 3 hops
splitstream⋆	99	99	95	89	83
splitframe⋆	99	99	94	84	84
iqzigzagidct⋆	99	99	92	89	83
mergestream⋆	99	99	91	90	85
mergeframe⋆	99	98	94	85	82
Trigger⋆	99	98	91	91	86
susan†	99	99	95	88	84
qsort†	98	94	85	84	81
toast†	99	98	95	85	84
untoast†	99	99	96	90	87
FFT†	99	98	95	90	86
bitcount†	99	98	94	87	82
basicmath†	99	98	92	88	84
adpcm†	99	99	94	90	85
LAME†	99	99	95	87	83
Matrix Multiplication‡	99	98	90	89	83
Producer‡	98	98	92	87	80

Table II
QoF OF IMPULSE RESPONSES. ⋆: MOTION-JPEG APPLICATION SPLIT IN 6 SUB-APPLICATIONS [32], †: MIBENCH EMBEDDED BENCHMARK [26], ‡: INTERNAL BENCHMARK.

VI. EXPERIMENTS AND RESULTS

Our approach was validated using Hotspot. For this purpose, the specification of $\mathcal{S}$ was taken from the Magma project, which provides a variety of multicore floorplans consisting of 2 core- through 64 core-layouts, see [31]. Each core is an appropriately scaled version of the Alpha 21264 processor. The knowledge of physical arrangement of cores on the chip multiprocessor is required, only if the user intends to apply approximations discussed in section V-B2. Such approximations were not made in our experiments. Although our technique does not require that all cores of $\mathcal{S}$ be homogeneous, the floorplans available in the Magma project consist of only homogeneous cores, and thus we report results for a multiprocessor system with homogeneous cores. Furthermore, no power traces were used, neither in the estimation of impulse responses, nor in the calculation of any temperature traces. To demonstrate scalability of our technique, a large floorplan consisting of an 8x8 arrangement of cores was chosen. The following sections describe results relating to the QoF of estimated impulse responses, as well as the accuracy of estimated temperature traces. Speedup due to our model, as compared to Hotspot is also presented. The time resolution t_s is 1 ms.

A. Accuracy of Estimated Impulse Responses

The order of the thermal model was limited to 10, at which the QoF achieved was greater than 90%. Further gains in QoF with increase in the order of the model were insignificant ($< 0.1\%$). The net effect of thermal resistance and thermal capacitance becomes increasingly complex, as the hop distance between two given cores increases. As a result, the QoF drops with the hop-distance from the active core. However, due to high lateral thermal resistance, the absolute error in temperature estimates small (5^0C). The results are summarized in Table II. Only the worst QoF per hop is reported for summary.

B. Speedup

A total of sixty mappings were evaluated, using applications from Table II. The scheduling policy used on each core was varied between EDF, LFF, RM and RR. For each mapping, temperature traces were calculated using our model, as well as Hotspot. The average time

Parameter(↓)	Our Model	Hotspot	Speedup
Mean Time (s)	24.853	29517	1187x
Maximum Time (s)	24.984	30057	1203x
Minimum Time (s)	24.795	27525	1110x
Standard Deviation (s)	0.054	547	

Table III
SPEEDUP ACHIEVED USING OUR APPROACH, AS COMPARED TO HOTSPOT.

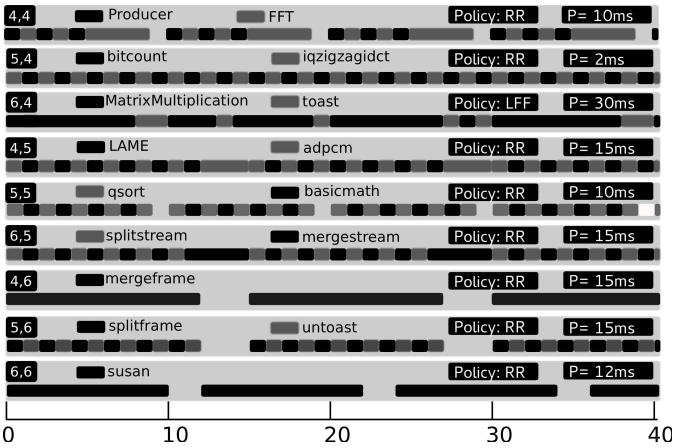

Figure 7. Schedule for nine cores. Time scale (ms) is indicated. Letter 'P' shows the period of each schedule.Scheduling policy for each core is also indicated.

taken for such calculations using our model was 24.9 seconds, while Hotspot averaged about 6 hours. The summary is provided in Table III. Further speedup is expected upon porting our algorithms to C/C++ from current Matlab/Java based environment.

C. Accuracy of Estimation

We consider a special mapping in which all applications are bound to cores located in a close spatial neighborhood. The temperature of each active core in this neighborhood is significantly influenced by all other active cores. Further, all active cores in this mapping are scheduled according to round-robin (1-ms quantum) policy, where possible, leading to significant number of context switches, and thus causing rapid variations in temperature over time. Such a mapping provides a good test for demonstrating the accuracy of temperature traces estimated using our technique.

The mapping is shown in figure 7. Taking the core with <5,5> as the center, applications are mapped on the immediate 1-hop neighborhood, totaling 9 heat generating cores. All other cores in this case are idle. The results are shown is shown in Figures 8. It is clear that the mapping in Figure 7 led to large changes in temperature on almost all active cores. Our thermal model was able to calculate correct temperature traces for all cores, well within the accuracy goal set up in the introductory section of this paper, see Figure 9.

In section I-A, the applications *producer* and *FFT* consume the same total power, but differ in their respective power density distributions. Both these applications were mapped onto core <4,4>, see Figure 7. It can be seen from temperature trace for core <4,4> in Figure 8 that producer and FFT applications produce distinctly different temperature affects. Our model was able to accurately capture the effect of differences in power density distribution between *producer* and *FFT*. Figure 10 shows a section of temperature trace for core <4,4> from Figure 8 for more clarity.

Other mappings, in which active cores are not immediate neighbors were also evaluated, and the prediction error was lower than what is reported in Figure 9. This is because an active core was not

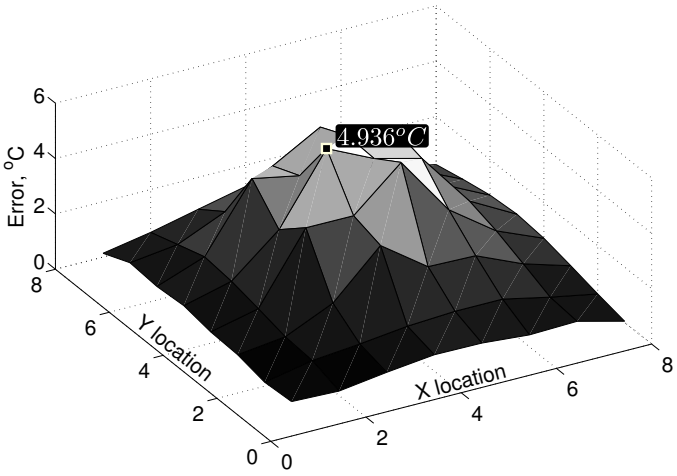

Figure 9. Maximum error in prediction over entire $\mathcal{S}$. Errors in absolute values.

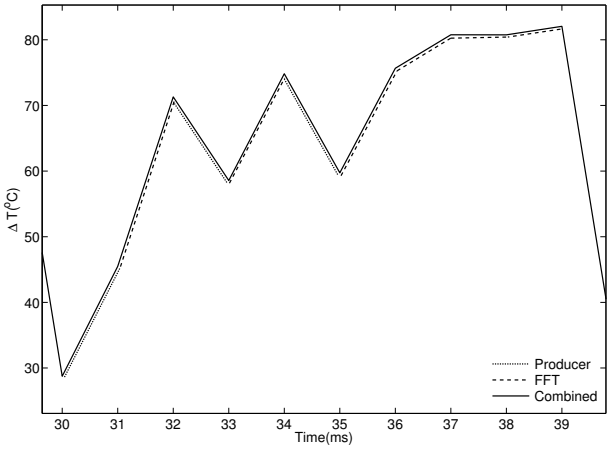

Figure 10. Section of temperature trace on core <4,4>. Both FFT and Producer applications have the same total power consumption, but lead to different temperature traces.

significantly influenced by its neighbors. Under these circumstances, estimation errors due to relatively lower QoF were limited by high thermal resistance, see Table II.

The speedup gained due to our approach allowed us to experiment with a lot of different mappings using the design space exploration loop. For instance, it was possible to reduce thermal cycles experienced by $\mathcal{S}$ by changing the binding of a few applications, see the new mapping in Figure 11. Notice that the total work done by each application remains unchanged. For instance, LAME executes for a total of 6ms, with a period of 15ms in both mappings. Also, a scheduling policy which minimized the number of context switches was chosen. See Figures 11 and 12. The overall error in estimated temperature is similar to the result shown in Figure 9; with the maximum error being 4.7^oC. It is not always possible to reduce thermal cycles by changing the bindings of applications, for a feasible scheduling policy for all cores may not exist.

VII. VARIATIONS AND OPTIMIZATIONS

It is not necessary to estimate impulse responses for all cores in the system, if the given system has a thermal symmetry. In this case, all cores are first classified into a set of thermally different locations (TDLs), see [22]. During the calibration step, applications need to be executed on only one distinguished core in each TDL. In the estimation stage, the calculation of thermal effect of an active core $\mathcal{S}_p$

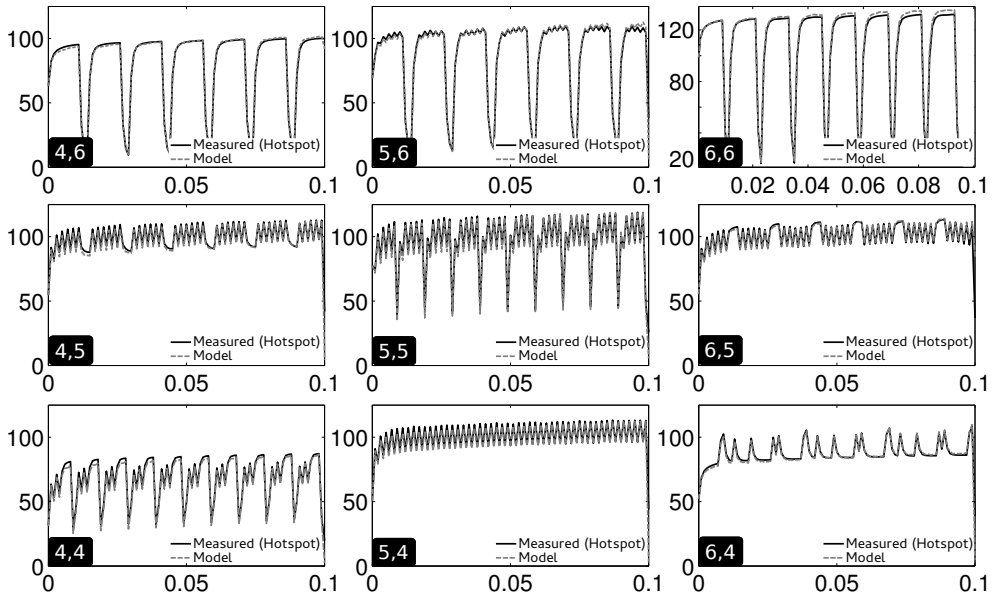

Figure 8. Measured temperature vs estimated temperature for nine active cores. Horizontal axes is time in seconds, vertical axes is $\Delta T(^oC)$. $\mathcal{V} = 47.7$.

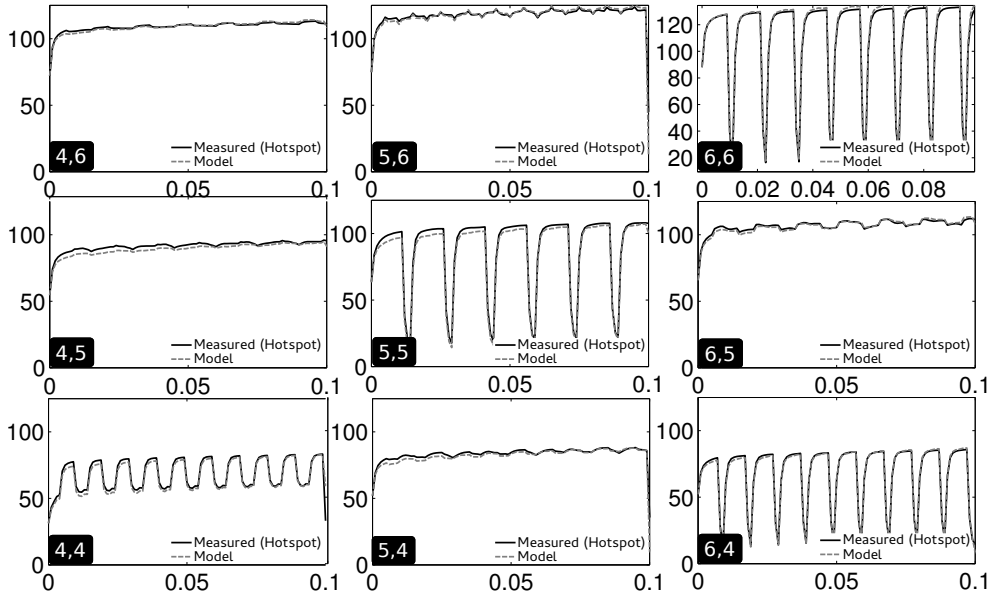

Figure 12. Reduction of thermal cycles by changing bindings of applications. Horizontal axes is time in seconds, vertical axes is $\Delta T(^oC)$. $\mathcal{V} = 25.1$

on core $\mathcal{S}_q$ proceeds in two steps. First, a sequence of transformations is determined which translates $\mathcal{S}_p$ to a core in one of the TDLs. Next, the same sequence of transformations is applied to core $\mathcal{S}_q$. This preserves the relative location of cores $\mathcal{S}_p$ and $\mathcal{S}_q$. Temperature trace calculation can now proceed normally. Thermal symmetry reduces the memory space required to store H, and a one-time computational burden required for calculation of H. The computational load for estimating the temperature trace for a given mapping may not change much, if large errors in temperature estimation are to be avoided, see section V-B2.

Also, the affect of caches on temperature has been implicitly factored in, during the impulse response estimation step. All caches were "clean" when an application $\mathcal{A}_i$ was executed on a core to collect associated temperature traces. Thus, even when multiple applications execute on the same core, the impulse responses already account for the effect of inevitable cache misses. In fact, the temperature traces estimated using our approach will be slightly higher than the measured temperature traces, making our approach thermally safe.

VIII. Conclusions

The paper presented a new calibration based approach for estimating accurate temperature traces. A compact thermal model was built using a small set of mappings and associated temperature trace measurement. The speed and accuracy of our approach enables exploration of a large set of candidate mappings using the design space exploration loop. The highlight of our approach is that it does not require any power-trace information, or the hard-to-obtain details about hardware, such as the detailed floorplan. Our technique can also account for differences in power densities on a core due to an application, even when the total power consumed by two or more applications is the same. This makes our technique applicable to any given set of embedded applications and hardware.

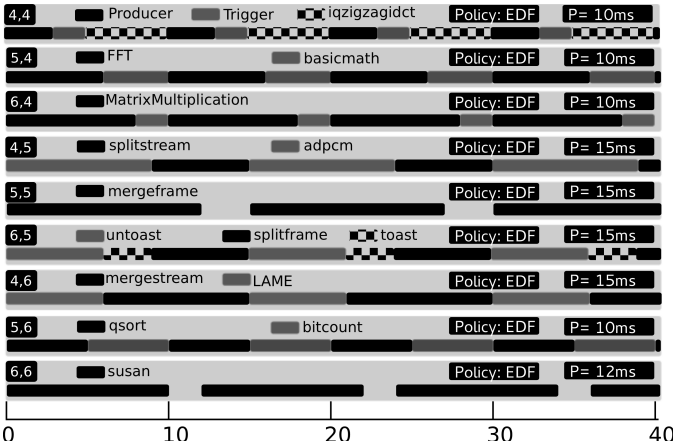

Figure 11. New mapping derived from the mapping in Figure 7.

REFERENCES

[1] C. Meenderinck and B. Juurlink, "(when) will cmps hit the power wall?" Delft University of Technology, Tech. Rep., 2008.

[2] E. Nowak, T. Ludwig, I. Aller, J. Kedzierski, M. Leong, B. Rainey, M. Breitwisch, V. Gemhoefer, J. Keinert, and D. Fried, "Scaling beyond the 65 nm node with finfet-dgcmos," in *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, sept. 2003, pp. 339 – 342.

[3] S. Park, J.-J. Chen, D. Shin, Y. Kim, C.-L. Yang, and N. Chang, "Dynamic thermal management for networked embedded systems under harsh ambient temperature variation," in *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, aug. 2010, pp. 289 –294.

[4] S. Schliecker, M. Negrean, and R. Ernst, "Response time analysis on multicore ecus with shared resources," *Industrial Informatics, IEEE Transactions on*, vol. 5, no. 4, pp. 402 –413, nov. 2009.

[5] C. Pinello, L. Carloni, and A. Sangiovanni-Vincentelli, "Fault-tolerant distributed deployment of embedded control software," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 5, pp. 906 –919, may 2008.

[6] A. Vassighi and M. Sachdev, "Thermal runaway in integrated circuits," *Device and Materials Reliability, IEEE Transactions on*, vol. 6, no. 2, pp. 300 –305, june 2006.

[7] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 314–324. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555794

[8] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, aug. 2007, pp. 38 –43.

[9] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off based on the ratio of off-chip access to on-chip computation times," in *Proceedings of the conference on Design, automation and test in Europe - Volume 1*, ser. DATE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 10 004–. [Online]. Available: http://dl.acm.org/citation.cfm?id=968878.969044

[10] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The impact of technology scaling on lifetime reliability," in *Dependable Systems and Networks, 2004 International Conference on*, june-1 july 2004, pp. 177 – 186.

[11] A. Ajami, K. Banerjee, and M. Pedram, "Analysis of substrate thermal gradient effects on optimal buffer insertion," in *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, 2001, pp. 44 –48.

[12] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor socs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 9, pp. 1127 –1140, sept. 2008.

[13] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, june 2003, pp. 2 – 13.

[14] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *Computers, IEEE Transactions on*, vol. 61, no. 4, pp. 563 –577, april 2012.

[15] N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy measurement and characterization with a case study of the arm7tdmi [microprocessors]," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 146 –154, april 2002.

[16] D. Rai, H. Yang, I. Bacivarov, J.-J. Chen, and L. Thiele, "Worst-case temperature analysis for real-time systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011, pp. 1 –6.

[17] Z. Wang and S. Ranka, "A simple thermal model for multi-core processors and its application to slack allocation," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, april 2010, pp. 1 –11.

[18] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *Proceedings of the 27th annual international symposium on Computer architecture*, ser. ISCA '00. New York, NY, USA: ACM, 2000, pp. 83–94. [Online]. Available: http://doi.acm.org/10.1145/339647.339657

[19] K.-J. Lee and K. Skadron, "Using performance counters for runtime temperature sensing in high-performance processors," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, april 2005, p. 8 pp.

[20] T. Wegner, C. Cornelius, M. Gag, A. Tockhorn, and A. Uhrmacher, "Simulation of thermal behavior for networks-on-chip," in *NORCHIP, 2010*, nov. 2010, pp. 1 –4.

[21] D. Li, S. X.-D. Tan, and M. Tirumala, "Architecture-level thermal behavioral characterization for multi-core microprocessors," in *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '08. Los Alamitos, CA, USA: IEEE Computer Society Press, 2008, pp. 456–461. [Online]. Available: http://portal.acm.org/citation.cfm?id=1356802.1356914

[22] J. Cui and D. Maskell, "High level event driven thermal estimation for thermal aware task allocation and scheduling," in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, jan. 2010, pp. 793 –798.

[23] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge, "Reliability modeling and management in dynamic microprocessor-based systems," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 0-0 2006, pp. 1057 – 1060.

[24] K. Skadron, T. Abdelzaher, and M. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, feb. 2002, pp. 17 – 28.

[25] H. Zumbahlen, Ed., *Linear circuit design handbook*. Newnes, 2008.

[26] D. o. E. E. University of Michigan and C. Sciences. Mibench version 1.0. [Online]. Available: http://www.eecs.umich.edu/mibench/

[27] Y. Hua and T. Sarkar, "Generalized pencil-of-function method for extracting poles of an em system from its transient response," *Antennas and Propagation, IEEE Transactions on*, vol. 37, no. 2, pp. 229 –234, feb 1989.

[28] L. Thiele, S. Chakraborty, M. Gries, and S. KÃŒnzli, "Design space exploration of network processor architectures," in *First Workshop on Network Processors at the 8th International Symposium on High-Performance Computer Architecture (HPCA8)*, Cambridge MA, USA, 2002, pp. 30–41.

[29] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," in *Proceedings of the 2004 annual ACM SIGAda international conference on Ada: The engineering of correct and reliable software for real-time & distributed systems using Ada and related technologies*, ser. SIGAda '04. New York, NY, USA: ACM, 2004, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1032297.1032298

[30] J. Wang and F.-y. Hu, "Thermal hotspots in cpu die and its future architecture," in *Intelligent Computing and Information Science*, ser. Communications in Computer and Information Science, R. Chen, Ed. Springer Berlin Heidelberg, 2011, vol. 134, pp. 180–185.

[31] S. V. Vinay Hanumaiah. (2009) The magma thermal simulator. Arizona State University. [Online]. Available: http://vrudhula.lab.asu.edu/magma/index.php

[32] P. Bourgos, A. Basu, M. Bozga, S. Bensalem, J. Sifakis, and K. Huang, "Rigorous system level modeling and analysis of mixed hw/sw systems," in *Formal Methods and Models for Codesign (MEMOCODE), 2011 9th IEEE/ACM International Conference on*, july 2011, pp. 11 –20.