Hot-Trim: Thermal and Reliability Management for Commercial Multicore Processors Considering Workload Dependent Hot Spots

Jinwei Zhang, *Student Member, IEEE*, Sheriff Sadiqbatcha, *Student Member, IEEE*, and Sheldon X.-D. Tan, *Senior Member, IEEE*

***Abstract*—This work proposes a new dynamic thermal and reliability management framework via task mapping and migration to improve thermal performance and reliability of commercial multicore processors considering workload-dependent thermal hot spot stress. The new method is motivated by the observation that different workloads activate different spatial power and thermal hot spots within each core of processors. Existing run-time thermal management, which is based on on-chip location-fixed thermal sensor information, can lead to suboptimal management solutions as the temperatures provided by those sensors may not be the true hot spots. The new method, called *Hot-Trim*, utilizes a machine learning-based approach to characterize the power density hot spots across each core, then a new task mapping/migration scheme is developed based on the hot spot stresses. Compared to existing works, the new approach is the first to optimize VLSI reliabilities by exploring workload-dependent power hot spots. The advantages of the proposed method over the Linux baseline task mapping and the temperature-based mapping method are demonstrated and validated on real commercial chips. Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that, compared to the existing Linux scheduler, core and hot spot temperature can be lowered by 1.15 °C–1.31 °C. In addition, *Hot-Trim* can improve the chip's electro-migration (EM), negative biased temperature instability, and hot-carrier-injection (HCI) related reliability by 30.2%, 7.0%, and 31.1%, respectively, compared to Linux baseline without any performance degradation. Furthermore, it improves EM and HCI-related reliability by 29.6% and 19.6%, respectively, and at the same time even further reduces the temperature by half a degree compared to the conventional temperature-based mapping technique.**

***Index Terms*—Lifetime reliability, multicore, post-silicon, power modeling, task allocation, thermal management.**

I. Introduction

POWER density increases with technology scaling, which can cause severe thermal and reliability problems in high-performance multicore systems [1]. Temperature and power has significant impacts on all major long-term reliability effects, such as electro-migration (EM) for interconnects, bias-temperature-instability (BTI), and hot-carrier-injection (HCI) for CMOS devices [2]. As a result, many research works have been investigated to find efficient methods to improve both system performance and reliability via dynamic thermal/reliability management (DTM/DRM) methods, which control the thermal and reliability behavior of multicore systems by online control such as task migration strategies [3], [4], [5], [6], [7].

However, existing DTM techniques either using dynamic voltage/frequency scaling (DVFS) or task migration are highly dependent on the on-chip location-fixed temperature sensors. Due to high design overheads, currently only a limited number of on-chip digital temperature sensors (DTSs) can be allocated on a silicon chip. A recent study shows that the number of hot spots on a typical commercial processor far exceeds the amount of embedded sensors [8]. Consequently, thermal and reliability management algorithms that solely depend on the sensors become insufficient for modern multicore systems, as power and thermal hot spots distinguish within cores under different workloads while having the same sensing temperature.

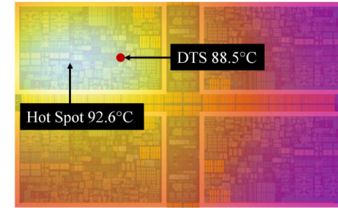

Fig. 1. Measured temperature of a hot spot versus the nearest sensor reading.

Fig. 1 shows a significant temperature difference[1] between a hot spot and the nearest sensor location on an Intel Core i7 quad-core processor under the SPLASH-2 workload *radiosity* (only displaying the quad-core area). Therefore, as the reliability of a core is mainly determined by the thermal hot spots, temperature per-core information alone is insufficient for DTM/DRM techniques. On the other hand, recent studies [9], [10] show that one can identify the power density distribution of a multicore processor with advanced thermal characterization.

[1]Temperatures are measured with a calibrated thermal imaging system (see Section IV-A).

Manuscript received 3 February 2022; revised 28 May 2022 and 11 September 2022; accepted 27 September 2022. Date of publication 21 October 2022; date of current version 20 June 2023. This work was supported in part by NSF Grant under Grant CCF-2007135, Grant CCF-2113928, and Grant OISE-1854276. This article was recommended by Associate Editor M. Zapater. *(Corresponding author: Jinwei Zhang.)*

The authors are with the Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA 92521 USA (e-mail: jzhan319@ucr.edu).

Digital Object Identifier 10.1109/TCAD.2022.3216552

1937-4151 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

Based on this observation, in this article, we introduce a new efficient and scalable task mapping algorithm for the thermal and reliability management for *commercial multicore processors* via machine learning-based modeling for power density at the true hot spots.[2] Our work is facilitated by an advanced thermal imaging system for measuring the spatial temperature across the full processor. Once temperature maps are measured, one can obtain the power density maps (the corresponding heat sources or hot spots) through the thermal-to-power technique using thermal measurements [10]. After that, we build a learning-based model for power density at the major hot spots in cores. We remark that the power or hot spot identification for commercial multicore processors under different workloads can also be carried out on chips with heat sink cooling in practical work settings [11].

The following summarizes the key contributions of this work.

1) We show that the existing task mapping techniques, which solely depend on per-core sensor temperature, may lead to subpar quality solution for chip reliability as the true hot spots of cores can be stressed unevenly.
2) Based on this observation, we employ a fast, run-time accurate machine learning model to estimate the exact spatial hot spots from the given workloads. With this, we propose a scalable and efficient task mapping approach to optimize the reliability of the multicore system.
3) Compared to existing works, the new task mapping approach is the first one to explore the workload-dependent power hot spots and its advantages over the existing Linux task scheduling method and temperature-based method, and has been demonstrated, validated on *real commercial multicore processors*. Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that, compared to the Linux baseline, the core and hot spot temperature can be reduced by 1.15 °C–1.31 °C. In addition, *Hot-Trim* can improve the chip's EM, negative BTI (NBTI), and HCI related reliability by 30.2%, 7.0%, and 31.1%, respectively, compared to Linux baseline without any performance degradation. Furthermore, it improves EM and HCI related reliability by 29.6% and 19.6% while further reduces the temperature by half a degree compared to the conventional temperature-based mapping technique.

This article is organized as follows: Section II reviews some related works. Section III discusses three major reliability effects and their models used in this work. Section IV presents the thermal imaging system setup and a motivation example for this work. Section V introduces the proposed hot-spot-aware task migration method. Section VI presents the results and comparisons on a real Intel i7 quad-core processor. Finally, Section VII concludes this article.

[2]In this article, hot spot is designated for power density hot spot instead of the traditional thermal hot spot. Power density hot spots are a superset of thermal hot spots and can be viewed as the potential thermal hot spots in general.

II. Related Work

Khdr et al. [12] introduced a multiobjective DTM method that aims to efficiently avoid thermal threshold violation and at the same time keeps the temperature balanced between cores based on the core temperature. It Derives a regression-based distributed temperature prediction model and a centralized task allocation model, it stops tasks that potentially cause overheating or imbalance of the cores, and resumes the tasks once there are available cores. Das et al. [13] developed a DTM technique that takes advantage of both the thermal profile within (intra) and across (inter) applications based on Q-learning, which learns the relationship between the task allocation, DVFS, and device aging/mean-time-to-failure (MTTF). Lu et al. [14] presented a task allocation method based on the core and router temperatures and predicts near-future temperature that assists the DTM. Their algorithm updates the prediction models after each allocation based on Q-learning. Q-learning-based control techniques are often subject to fast-rising learning spaces as the states and actions of systems expand. Iranfar et al. [15] proposed a machine learning or ML-based power/thermal management approach that uses a heuristic to limit the learning space by assigning a specific set of available actions to each existing state. A recent state-of-the-art DVFS technique enables scaling down of the management cycle to a microsecond time scale and achieves fast per-core DVFS [16], which significantly reduces the power consumption across cores. Recently, Zhang et al. [17] proposed a deep reinforcement learning-based method to allocate the tasks based on the hot spot power rather than temperature information, which infers the power information has great potential to be used to improve the system and thermal performance of the chip.

III. Reliability Models

In this section, we briefly review the three major VLSI reliability effects: 1) the EM for interconnects; 2) the NBTI; and 3) HCI for MOSFET devices and their calculation models. We note that the proposed method can consider other failure effects as well. The three failure effects are the dominant aging effects in the VLSI systems as EM will cause the power grid network to be time varying and changes the voltage drop over time. NBTI and HCI can lead to the threshold voltage shift such may cause failure to signal transition and timing. In addition, calculations for the aging and lifetime due to EM and NBTI are implemented through an open source tool—LifeSim [18], which we will explain in detail in Section VI.

A. EM Model

The currently employed method of predicting the time to failure regarding the EM effect is based on a physics-based EM analysis method [19], [20]. It comprehensively models the EM effect considering the void nucleation phase and growth phase during which the wire resistance starts to change. Specifically, the void nucleation time can be expressed as follows:

$$t_{\text{nuc}} \approx \tau^* e^{\frac{EV}{kT}} e^{\frac{f_V \Omega}{kT}} \left(\sigma_{\text{res}} + \frac{eZ\rho l}{4\Omega} j \right) \text{In} \left\{ \frac{\frac{eZ\rho l}{4\Omega} j}{\sigma_{\text{res}} + \frac{eZ\rho l}{4\Omega} j - \sigma_{\text{crit}}} \right\} \quad (1)$$

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

with $\tau^* = (l^2/D_0)e^{(E_D/kT)}(kT/\Omega B)$. Here, E_V and E_D are the activation energy of vacancy formation and diffusion, f_v is the ratio of volumes occupied by vacancy and lattice atom, and σ_{res} and σ_{crit} are the residual stress and critical stress. Ω is the atomic volume, l is the wire segment length, eZ is an effective charge of the migrating atoms, j is current density, T is temperature, and ρ is the wire electrical resistivity.

At the system level, to model the current density, we follow the similar formula used in the RAMP [21] and the work in [22], which can be related to the switching probability of the line, α, as follows:

$$j = \frac{CV_{dd}}{WH} \times f \times \alpha \tag{2}$$

where C, W, and H are the capacitance, width, and thickness, respectively, of the line and f is the clock frequency.

Once the void is formed in the wire it starts to grow and the wire resistance increases over the time. The drift velocity of the void edge is expressed as follows:

$$\theta = \frac{D}{kT} eZ\rho j. \tag{3}$$

Further, kinetics of the wire resistance change with respect to the growth time is approximated as follows [19]:

$$\Delta r\left(t_{\text{grow}}\right) = \theta t_{\text{grow}}\left[\frac{\rho_{Ta}}{h_{Ta}(2H+W)} - \frac{\rho_{Cu}}{HW}\right] \tag{4}$$

where ρ_{Ta} and ρ_{Cu} are the resistivity of the barrier material and copper, W is the line width, H is the copper thickness, and h_{Ta} is the barrier layer thickness. The growth time is calculated for a given resistance percentage change threshold (such as 10%). The final time to failure due to EM effects is determined by adding the nucleation time and the void growth time together.

B. NBTI Model

NBTI occurs when negative biased voltage is applied to the gate of a PMOS transistor, the presence of holes in the channel causes Si-H bonds to break at the interface between the gate oxide and the channel, causing positive traps in the interface, which increase V_{th} [23]. The reaction rate mainly depends on the temperature T and the supply voltage V_{dd}. The model of lifetime reliability due to NBTI we use is based on the work by Srinivasan et al. [24]. MTTF due to NBTI at a temperature T, is given by

$$\text{MTTF} \propto \left[\left(\ln\left(\frac{A}{1+2e^{\frac{B}{kt}}}\right) - \ln\left(\frac{A}{1+2e^{\frac{B}{kt}}} - C\right)\right) \times \frac{T}{e^{\frac{-D}{kt}}}\right]^{\frac{1}{\beta}} \tag{5}$$

where A, B, C, D, and β are fitting parameters using the published NBTI failure data [25], and k is the Boltzmann constant. Based on the model in [24], the values we use are $A = 1.6328$, $B = 0.07377$, $C = 0.01$, $D = -0.06852$, and $\beta = 0.3$.

C. HCI Model

HCI refers to the high energetic carriers, which is the result of high electric fields in the drain region of a transistor, are injected into the gate oxide. These carriers form interface states and eventually result in performance degradation (increase of V_{th}) in the transistor under stress [26]. The equation below evaluates the HCI-induced threshold voltage increase [27]

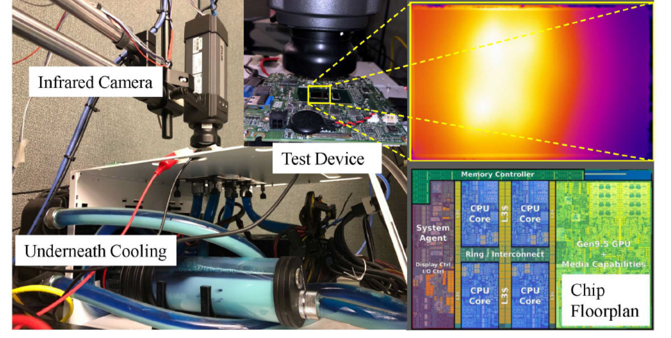

Fig. 2. Infrared thermography system.

$$\Delta V_{\text{th}}(\alpha, T, V_{dd}, t) = A_{\text{hci}} \cdot u(V_{dd}) \cdot v(T) \cdot \sqrt{\alpha \cdot f \cdot t} \tag{6}$$

with

$$u(V_{dd}) = e^{\left(\frac{V_{dd}-V_{\text{th}}}{E_1}\right)},\ \ v(T) = e^{\left(-\frac{E_a}{kT}\right)} \tag{7}$$

where t stands for operation time, α is activity factor, and f is core frequency. In addition, t_{ox} is the oxide thickness, and E_1 depends on the device specifications, temperature, and V_{dd}. Further, A_{hci} is a technology-dependent constant and activation energy E_a is considered a positive constant.

D. Summary of Reliability Models

In summary, EM causes the power grid network to be time-varying and changes the voltage drop over time. NBTI and HCI lead to the threshold voltage shift such that may cause failure to signal transition and timing. In this work, we set the failure criterion to be 10%, i.e., 10% wire resistance change due to EM and 10% change of threshold voltage due to NBTI and HCI are considered end of lifetime.

IV. Observation and Motivation

The analysis, measurements, and implementations of this work are all based on real systems. The reason is measuring from a real processor when it is executing workloads is more precise and has more realistic meaning than from computer simulators. Second, open-source computer simulators hardly include the ready-to-use architecture resources for the latest off-the-shelf processors.

A. Thermography System Setup

In order to acquire precise thermal and power information within the core, a proper measurement system for spatial temperature is critical. To this end, we have adopted the thermography measuring system proposed in [28]. This setup features a thermoelectric device mounted on the other side of the motherboard right beneath the processor allowing it to be cooled from underneath, as opposed to heat sinks drawing heat upward. This setup leaves the front side of the processor fully exposed to the infrared camera without any interference layer in-between, as shown in Fig. 2. An adjustable dc power

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

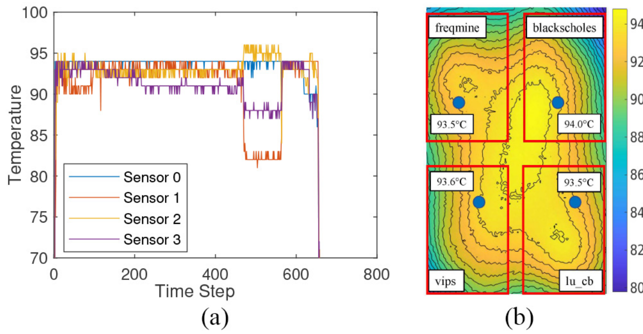

Fig. 3. (a) On-chip sensor readings (one sensor per core). (b) Measured temperature at the sensor locations (blue dots).

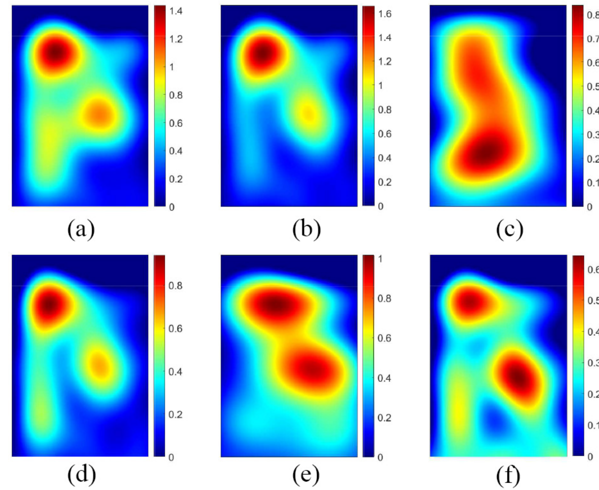

Fig. 4. Power patterns of PARSEC-3.0 and Splash-2 benchmark workloads on a real Intel Core-i7 processor at the core scale (within the core). (a) Radiosity. (b) Streamcluster. (c) ×264. (d) Ferret. (e) Raytrace. (f) Canneal.

supply is used to control the heat flow through the thermoelectric device so that the operating conditions can be matched to the baseline cooling unit (stock heat sink) using the calibration method discussed in [28]. Unlike the traditional flowing-oil-based front-cooling methods [29], no decoupling procedures are required in this setup. The thermal image capturing rate can reach as high as 60 frames/s.

B. Glance of Hot Spots

We first illustrate how the cores can be stressed in various ways that the sensors cannot tell. Then, the idea of optimization over the existing techniques will be described at a high level in this section.

Fig. 1 shows the measured spatial temperature (Intel Core-i7 quad-core) when it is under a workload (Splash-2 benchmark *radiosity*). It reveals that the temperature between a true hot spot and the nearest sensor can be quite different. When there are many cores under workloads, temperature sensors are likely to measure the same or similar temperature even though cores are under different workloads being stressed in different patterns. We measured the temperatures in the time axis by the embedded sensors, shown as Fig. 3(a). It is obvious that temperatures across all sensors, at least two or three, are often very close during the runtime. Note that the precision of sensors is only integer. Moreover, as shown in Fig. 3(b), when four workloads (*lu_cb*, *vips*, *blackscholes*, and *freqmine*) are running on the four cores, respectively, the temperatures at sensor locations measured by the imaging system are 93.5 °C, 93.6 °C, 94.0 °C, and 93.5 °C, where the difference is quite small.

We remark that the thermal hot spots are always the power density hot spots or the heat-source hot spots. But this is not true the other way around as shown in a recent study [8]. Heat-source hot spots can be viewed as *potential thermal hot spots* or their *spatial distributions*, which can be activated by specific workload. The hot spots from heat sources or power sources can provide more useful, especially critical information about the true thermal hot spot distributions for real commercial multicore processors, which is the motivation in this work to use power density hot spots. In the sequel, for the sake of simplicity, hot spot simply means power density hot spot and power or power pattern means the power density or power density pattern.

We calculate the core's power patterns of various PARSEC-3.0 and Splash-2 workloads and some typical patterns are shown in Fig. 4 as examples, respectively. There are three primary hot spot locations observed in the core. Some workloads have higher and sharper power peaks than others, while other workloads show more even power distribution. Consequently, utilization of the hardware resources, reflected by the power density at hot spots, indicates the different stresses of the silicon chip. Hence, there is a considerable potential for task migration operations to optimize the thermal and reliability performance by utilizing the hot spot power information.

For illustration, the typical power density measured at the hot spots with respect to workloads are listed in Table I, where the three primary hot spots are named as HS1, HS2, and HS3 are listed. It should be noted that the applications may contain both serial and parallel threads, and the power density values listed in Table I are averaged values through the thermal-to-power calculation when the applications run into a thermal steady state, hence the parallel phase (also dominant phase) of the application is considered in this table. The measuring workflow can be implemented on other chips as well.

V. Proposed Hotspot-Aware Task Allocation Framework

In this section, we will describe the overall workflow for the proposed task allocation algorithm. The framework consists of two major components: 1) a detector model detecting the power density of the primary hot spots and 2) a management controller that collects the power information of those hot spots of all the cores and controls the allocation of threads. For the sake of comparison, we will not interfere with the DVFS policy of the system.

A. Learning-Based Hot Spot Modeling and Detection

One important aspect of the proposed method is to know which hot spot locations are active or invoked by the workload in a core in real time. This can be achieved by using deep

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

TABLE I
AVERAGE POWER DENSITY (W/mm^2) AT HOT SPOTS FOR VARIOUS WORKLOADS

Workload	HS1 Power	HS2 Power	HS3 Power
blackscholes	1.04	1.56	1.82
bodytrack	0.92	1.40	1.82
fluidanimate	0.75	1.3	1.8
streamcluster	0.52	1.06	1.57
dedup	0.75	1.0	1.56
facesim	0.75	1.0	1.56
swaptions	0.52	1.0	1.53
lu_cb	0.52	1.0	1.52
freqmine	0.52	0.91	1.38
radiosity	0.72	1.1	1.37
vips	0.26	1.0	1.3
radix	0.52	1.0	1.2
ferret	0.52	0.65	0.9
canneal	0.39	0.63	0.6
raytrace	0.56	1.08	1.15
x264	0.82	0.26	0.75
fft	0.5	0.9	1.3
ocean_cp	0.26	1.0	1.43
volrend	0.78	1.0	1.3

neural networks (DNNs). We estimate the power density at the hot spots of the off-the-shelf multicore processors during real time from the online utilization metrics. Specifically, we implement a DNN as a supervised learning model which can estimate the power densities at hot spots in cores from the underlying real-time resource utilization information.

In our implementation, we take advantage of a multilayer perception (MLP) network with two fully connected layers and a dropout layer for hot spot power density detection (Fig. 5). The input data for the network's training and inference is obtained from Intel's Performance Counter Monitor (IPCM) [30], IPCM provides the system-level utilization metrics that we will be utilizing in this work. For non-Intel chips, the equivalent performance monitors can be used (i.e., AMD uProf [31]). IPCM provides the real-time processor package and core-wise performance metrics, such as frequency, energy, instruction per cycle, cache hit, read/write rate, etc., as well as the sensed temperature from the embedded sensors. The Intel chip used in this study, i.e., Core i7-8650U, has four cores and each core supports two threads with Intel's hyperthreading technology. Table II shows the complete list of IPCM performance metrics from both the package and core-wise (or thread wise) domains that are used in this work. We note that the IPCM-based full-chip thermal map modeling method has been proposed recently [32]. There are 30 metrics corresponding to the whole package domain, and 16 metrics for each core thread. Considering that hyperthreads may happen on this chip, when measuring the training data we disabled the hyperthreading option, having one core only execute one thread at a time instead of two. In this way, we make sure the externally captured thermal images are matched for the thread executed in the core. Otherwise, the thermal images and the following calculated power densities would be a contribution of two separate threads running concurrently on the same core due to the hyperthreading function. Once the NN model is trained it

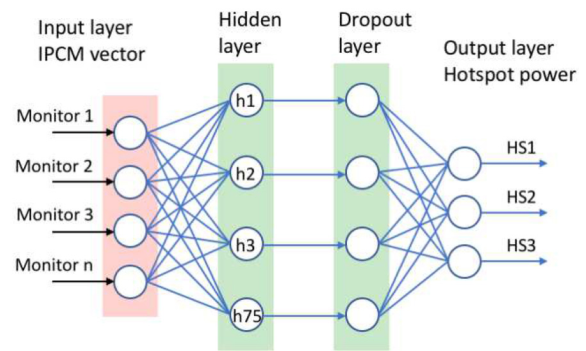

Fig. 5. Power detector network architecture.

TABLE II
HIGH-LEVEL PERFORMANCE METRICS (INTEL PCM)

Package			Core	
Exec	Read	C1res%	Exec	c0res%
IPC	Write	C2res%	IPC	c1res%
Freq	INST	C3res%	Freq	C3res%
AFreq	ACYC	C6res%	Afreq	C6res%
L3Miss	Time	C7res%	L3Miss	C7res%
L2Miss	PhysIPC	C8res%	L2Miss	T_{sens}
L3Hit	PhysIPC%	C9res%	L3Hit	
L2Hit	INSTnom	C10res%	L2Hit	
L3MPI	INSTnom%	Energy(J)	L3MPI	
L2MPI	C0res%	T_{sens}	L2MPI	

can be used in a thread-wise manner as one core's power is a combination by two threads. In total, the input vector contains 46 IPCM metrics for the core-wise (or thread wise) hot spot power density detection neural network. In our later experiments, we limit one core to execute only one thread in order to reach easier software implementation of the algorithm in the user space, which will not lose the validity of the algorithm.

Output data of the network are the power densities at the identified primary hot spots of the core in real time. In our case, the output dimension is three due to three identified hot spots. Note that the name of the workload is not a factor in the power detector network. We obtain the core's hot spot power densities by deploying a recently proposed thermal-to-power transformation approach [8]. The corresponding thermal imaging measurements are collected at the same time when the processor is under workload. IPCM tool is launched also at the same time when the processor is under workloads, data of the performance counter metrics is sampled at the same frequency and synchronized to the thermal image capturing. Then, spatial power patterns are calculated through thermal measurements and power densities at the primary hot spots are extracted [10]. Finally, IPCM metric vectors serve as inputs and power densities of hot spots per core serve as targets for the learning-based network. We measured 7200 thermal images with the highest camera frequency (60 Hz) and the synchronized IPCM metric vectors corresponding to each workload application, where 20 applications from PARSEC-3.0 and SPLASH-2 are measured for the network training and test procedure. In our study, we observed that the power patterns of all the workloads are steady during almost their entire execution time except for

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

a slight instant fluctuation at the beginning. Moreover, the same workload demonstrates the same power pattern across different cores when executing on multiple cores parallelly. This convention actually shrinks the complexity of model learning and makes the network easy to train and use. We will present the inference (detection) accuracy of the online power density detector in the results section (Section VI).

B. Task Management Controller

The task management controller collects power information of hot spots, maintains the core and task status, and allocate incoming or ongoing tasks. We define the following concepts for a clear description.

Task Queue: Incoming tasks/applications are put in a queue following the first-in-first-out (FIFO) order. It is assumed there is no priority order among them since the priority is not related to this study.

Parallelism Count: The number of parallelisms is usually determined by the user space. To generalize the new algorithm for tasks running with multiple parallel threads, each element in the task queue contains the name of task and the number of parallel threads it asks for. In our implementation, the task will be assigned with as many available cores as the user-determined parallelism count by setting the task's CPU affinity, where CPU affinity means a list of cores the task can run on. Note that we only set/update the task's CPU affinity in every management cycle instead of assigning the underlying specific threads to the specific cores. The order of threads is maintained by the task itself and the functionality is guaranteed.

Core Status: Cores have two status, either available or busy.

Waiting Parallelism Queue: For an incoming multithreaded task that requests multiple cores for parallel execution, the number of available cores may be less than the number it requests for. Then, all the available cores are assigned to the task and the excessive number of parallelisms requested is put in the waiting parallelism queue till other cores become available.

Management Cycle: Threads of tasks are migrated among cores every management cycle, δt, e.g., 1–5 s.

Sampling Interval: Every sampling interval, e.g., 100–1000 ms, the management controller updates the core, task, and hot spot status that it maintains, and allocates the queued task to cores immediately once there are available cores detected.

Fig. 6 illustrates an example of the task queue, waiting thread queue, instant mapping, and the corresponding power pattern on a quad-core processor. In this example, task 1 first occupies two cores, then task 2 occupies one core. Task 3 requesting for two cores is only mapped to one core given only one core left available. Task 3's another thread request is held in the waiting thread queue for the next available core. In our example, the processor layout follows a central symmetric pattern.

It should be noted that to reduce the complexity of interfering with the OS scheduler in this work, the management controller checks the status at each sampling interval from the user space rather than the kernel space of the OS.

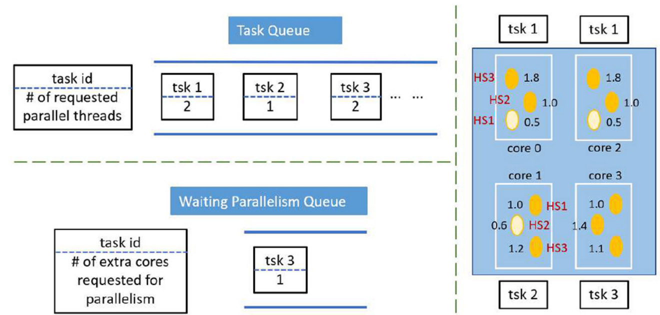

Fig. 6. Example of task queue, waiting thread queue, and the initial instant mapping.

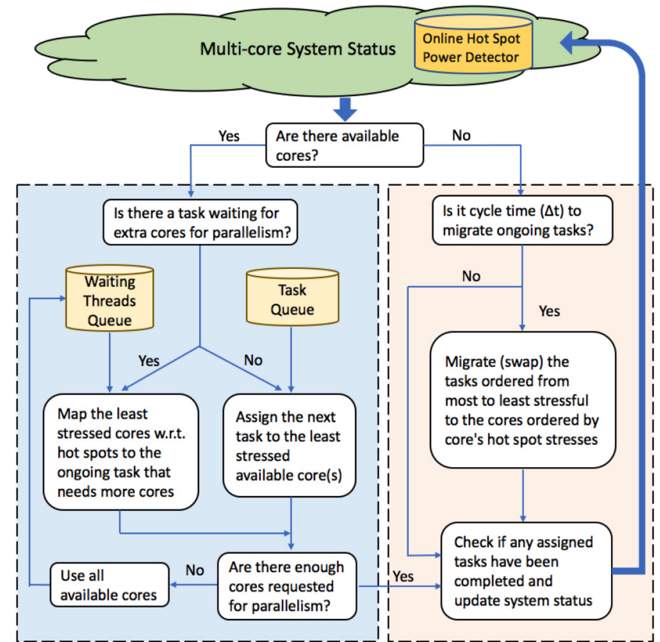

Fig. 7. Task management workflow.

In the future, once the technique has been built into the OS kernel, the model does not need to check the status using the interval manner anymore, it should know those events immediately instead. We also comment that the power detector model does not need to calculate the hot spot power density all the time. As discussed, the power pattern of the same task is quite steady on the time axis. Hence, the hot spot power information can be sampled, stored, and reused. If an unknown task comes, the detector model will wake up for a short period of time intermittently and obtain an averaged hot spot power data for that task. In this way, the computation cost by the power detector model is much shrunk.

Fig. 7 presents the workflow of the management controller. At the top, the model accesses the multicore system information it needs, including the core status, queue status, and hot spot powers. In each information sampling cycle, it first checks if there are available cores. When yes, it then checks the waiting thread queue to see if the ongoing task needs more cores. It always allocates the waiting thread before the next task unless the waiting threads queue is empty (i.e., FIFO). When available cores are not enough for an incoming

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Task Migration in a Management Cycle

Input: M cores, N ongoing tasks, H hot spots per core, current task map and performance counter metrics (IPCM)
Output: New task map *newMap*

1: $curMap \leftarrow [tsk_{1,1}, \ldots, tsk_{i,m}, \ldots, tsk_{N,M}]$

2: **for** i = 1 to N **do**
3: $\quad P_{tsk,i} \leftarrow [p_1, \ldots, p_j, \ldots, p_H]_i = Net(IPCM(tsk_{i,m}))$
4: $\quad P^{\max}_{tsk,i} \leftarrow \max(P_{tsk,i})$
5: $\quad h^{wst}_{tsk,i} \leftarrow argmax(P_{tsk,i}),\ 1 \leq h^{wst}_{tsk,i} \leq H$
6: **end for**
7: $maxPwrs \leftarrow [P^{\max}_{tsk,1}, \ldots, P^{\max}_{tsk,i}, \ldots, P^{\max}_{tsk,N}]$
8: $wstHSs \leftarrow [h^{wst}_{tsk,1}, \ldots, h^{wst}_{tsk,i}, \ldots, h^{wst}_{tsk,N}]$
9: $sortedTsk \leftarrow \text{argsort}_{tsk}(maxPwrs, \text{reverse} = true)$
10: Sort *wstHSs* by the same order to match the tasks in *sortedTsk*

11: Initialize $Cores \leftarrow set\{1, 2, \ldots, M\}$
12: Initialize $newMap \leftarrow [None_1, \ldots, None_m, \ldots, None_M]^*$
13: **for** i = 1 to N **do**
14: $\quad tsk_i \leftarrow sortedTsk[i]$
15: $\quad h \leftarrow wstHSs[i]$
16: $\quad P_{c,h} \leftarrow [p_{(c,1)}[h], \ldots, p_{(c,m)}[h], \ldots, p_{(c,M)}[h]]$
17: $\quad prefCoreLst \leftarrow \text{argsort}_c(P_{c,h})$
18: $\quad$ Initialize $mappedCores_{tsk,i} \leftarrow set\{\ \}$
19: $\quad$ **for** *core* in *prefCoreLst* **do**
20: $\quad\quad$ **if** *core* in *Cores* **then**
21: $\quad\quad\quad$ add *core* to $mappedCores_{tsk,i}$ for tsk_i
22: $\quad\quad\quad$ remove *core* from *Cores*
23: $\quad\quad$ **end if**
24: $\quad$ **end for**
25: $\quad$ Update $newMap \leftarrow mappedCores_{tsk,i}$
26: **end for**
27: Use *newMap* for task migration operation

task from the task queue, the task will be mapped to all the available cores and registered to the waiting thread queue for future available cores.

In every management cycle, the controller migrates the ongoing tasks from cores to cores according to the proposed mapping algorithm, which will be discussed in the next section. Afterward, the controller updates the system status it maintains.

C. Proposed Mapping Control Algorithm

As we already observe that the power (density) at the hot spots can vary considerably depending on the specific workloads. The higher power peaking at the hot spot, the more severe threat to the core's reliability. And the longer time the hot spot has been stressed, the lower reliability, too. Therefore, we develop a heuristic mapping algorithm that allocates tasks such that the average power peaking at the hot spots is mitigated. The mapping algorithm deals with two scenarios, one is migrating the ongoing tasks among cores, and the other is mapping the waiting threads or an incoming task to the available cores.

TABLE III
EXEMPLARY ORDERING OF TASKS AND CORES AND MIGRATION

Task Order	Worst Hot Spot	Preferred Cores	Mapped Cores
1) Tsk 1	HS3: 1.8 $\mathrm{W/mm^2}$	1, 3, 0, 2	1, 3
2) Tsk 3	HS2: 1.4 $\mathrm{W/mm^2}$	1, 0, 2, 3	0
3) Tsk 2	HS3: 1.2 $\mathrm{W/mm^2}$	1, 3, 0, 2	2

1) Migrate the Tasks: Suppose the processor has M cores where each core has H primary hot spots. And the current task map corresponding to the ongoing N ($N \leq M$) tasks is noted as $[\text{tsk}_{1,1}, \ldots, \text{tsk}_{i,m}, \ldots, \text{tsk}_{N,M}]$, where $\text{tsk}_{i,m}$ means the ith task running on the mth core and can be *None* if no task runs on that core. One task may run on multiple cores. Power at all H hot spots of a core activated by the task tsk_i is noted as $P_{\text{tsk},i} = [p_1, \ldots, p_j, \ldots, p_H]_i$, where p_j means the power at the jth hot spot activated by tsk_i. The proposed task migration algorithm is elaborated in Algorithm 1. We first estimate the power at hot spots activated by every running task (lines 2 and 3) through the machine learning-based power detector. And find the maximum power $P^{\max}_{\text{tsk},i}$ of hot spots (line 4) and the worst hot spot $h^{\text{wst}}_{\text{tsk},i}$ (line 5) activated by that task. Then, we sort the tasks by how stressful they are by the maximum power of hot spots they activate (lines 7 and 9). The task having a higher maximum power of hot spots is considered more stressful. If two tasks stimulate the same maximum power (not necessarily on the same hot spot), then compare their second highest hot spot power, and so on so force. For example, according to the data shown in Table I, *blackscholes* should be ordered ahead of *bodytrack*, then *fluidanimate*. Then, similarly, for each task the cores are ordered from the most preferred to least preferred (*prefCoreLst*) with respect to that task (lines 13–17). Here, h indicates the worst hot spot that will be stressed by this task most and $p_{(c,m)}[h]$ denotes the accumulated power (energy) at the hot spot h of the core m. Lines 19–24 map the task based on the order of its preferred cores. The task which is more stressful is taken care of earlier as having higher priority to pick the preferred cores than the less stressful tasks. If some preferred cores are already scheduled for other tasks in this management cycle, then these cores will be skipped for this task.

Once the new task map has been obtained, the task management controller migrates the tasks for this cycle. Following the example in Fig. 6, Table III shows the order of the tasks, the order of their preferred cores, and the newly mapped cores, respectively. Fig. 8(a) further illustrates the resulted mapping diagram for the management cycle.

2) Map the Waiting Threads: This is similar to migrating the tasks. Locate the hot spot the targeted task will stress most and order the cores by accumulated hot spot power at that location. For example, if we are to allocate threads of *canneal*, then the cores should be ordered by their HS2 power in the management cycle because HS2 is the worst hot spot stimulated by *canneal*. Following the example shown in Figs. 6 and 8(b) shows mapping a waiting thread of task 3 to the best available core, core 1, when cores 0, 1, and 2 become available simultaneously.

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

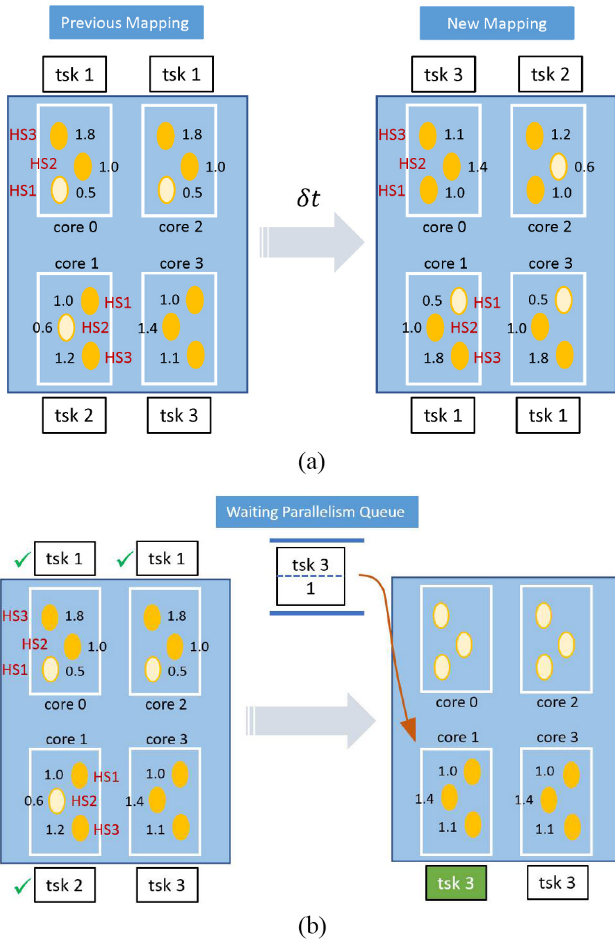

Fig. 8. (a) Task migration in a management cycle. (b) Example of mapping a waiting thread to the available cores.

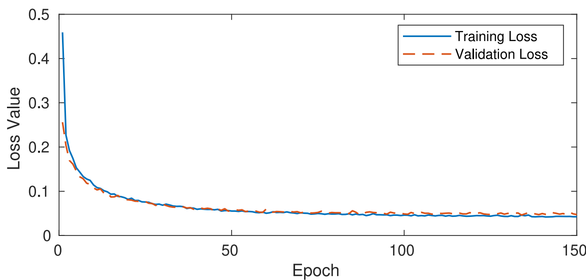

Fig. 9. Power density detector neural network learning curves.

VI. Experimental Results and Discussions

In this section, we present the results for the proposed hotspot-aware task control method, Hot-Trim, for thermal and reliability management of multicore processors. We implement and validate our method on a commercial Intel i7-8650U processor that features 4-CPU cores with PARSEC-3.0 and SPLASH-2 benchmark workloads [33], [34] (we write the benchmark workloads as tasks to be brief in this article).

First, we present the performance of the power density detector neural network. We measured 7200 thermal images with the highest camera frequency (60 Hz) and the synchronized IPCM metric vectors corresponding to each workload

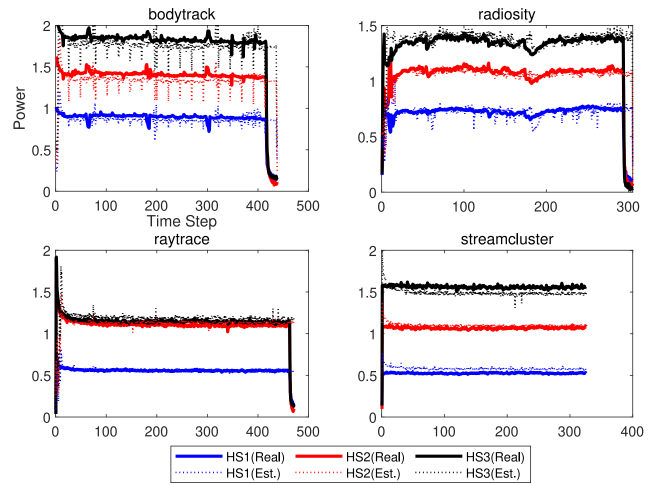

Fig. 10. Power density (W/mm^2) versus time steps (60 Hz) for workloads. Estimated power density compared to the measured power density at the identified hot spots.

application, where 20 applications from PARSEC-3.0 and SPLASH-2 are measured for the network training (80% data) and test procedure (20% data). 20% of training data is used for validation during the training procedure. Fig. 9 shows the training loss and validation loss during the training procedure of 150 epochs. We mark that the specific configuration of the MLP network (# of nodes, # of layers, etc.) is not an exact science. In this work, we used one hidden layer with 75 nodes and a dropout layer with a 0.5 ratio between the input and output layers, and the learning rate is 0.0005. We did not observe overfitting on the trained network model. Fig. 10 illustrates the comparison between the estimated power density and the measured power density traces at the identified hot spots, where the estimated power density is obtained from the learning-based power density detector neural network and the measured power density is obtained through the thermal-to-power method [8], [10]. As we can see, the estimated power traces align quite well with the real measured power traces. It should be noted that in the training procedure, thermal and IPCM data is obtained with the highest camera frequency. Whereas in the following management experiments, the cycle period is chosen as 2 s and IPCM sampling interval is 200 ms. In our experiments, we observe that the power pattern of the running task usually takes only 200–300 ms to become steady after launching, as examples shown in Fig. 10. The migration and sampling frequencies are relatively low but sufficient. In this work, we obtain the power density estimation from the IPCM once at the end of every management cycle, i.e., before the next migration, and average the power density estimations after every management cycle of the task and average between cores if running on multiple cores. The computation overhead is reasonably low such that the online inference time is less than 100 μs and the overall computational time regarding the whole Algorithm 1 in one management cycle is between 300 and 400 μs. We will present more details in the next section.

Second, we compare the performance of the proposed Hot-Trim with existing mapping methods. In this work, we compare three methods, i.e., Linux baseline mapping, temperature-based mapping, and the proposed Hot-Trim mapping in terms

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
TEST CASES OF TASK SERIES

Test Cases	Task Series	Input Size
Case 1	[(ferret, 4), (streamcluster, 2), (canneal, 2), (raytrace, 2), (bodytrack, 1), (lu_cb, 2), (radix,2), (dedup, 2), (fft, 2), (vips, 1), (facesim, 2), (freqmine,1), (fluidanimate, 2), (bodytrack, 2), (ferret, 4)]	Large
Case 2	[(freqmine, 2), (blackscholes, 1), (dedup, 1), (canneal, 2), (radix, 2)]	Native
Case 3	[(vips, 2), (blackscholes, 1), (dedup, 1), (radix, 1)]	Native

of runtime performance, thermal behavior, and the three critical reliabilities as mentioned earlier. Specifically, the Linux baseline mapping means when allocating the tasks, tasks will be launched without assigning the CPU core affinities. We let the OS scheduler choose the CPU cores automatically to execute the tasks. For temperature-based mapping, we implement the most popular greed-based mapping policy such that the task is always mapped or migrated to the coolest core based on the thermal measurements of on-chip sensors. If there are multiple tasks executing on multiple cores, the tasks executing on the hot cores will be migrated to the cooler cores. Each mapping method will be deployed to execute the same series of tasks. In the meantime, performance counter metrics and thermal images of the full chip will be captured to investigate the runtime performance, thermal behavior, and reliabilities. To make sure the comparison is comprehensive, we have gone through a few different experiment scenarios.

A. Comparison in System Performance

First, we start by investigating whether the proposed method degrades the runtime performance and how it compares to the Linux baseline, in other words, whether the total execution time is prolonged. If it degrades the original performance seriously then there would be no sense to propose more. The Linux kernel version on the test processor is 5.0.9-301.fc30.x86, and the OS distribution is Fedora 30. Note that in this work we only deploy the task mapping policy but not the DVFS scheduling, instead, we let the OS handle the DVFS as it normally does. First, we compose diverse task series that contain various numbers and types of tasks. Each element in the task series is presented as (*task name*, *# of threads needed*). We also deploy two different input dataset size, *Large*, and *Native* for the tasks in the PARSEC-3.0 and SPLASH-2 benchmarks. The user time of tasks with *Large* input size usually lasts for about a few seconds to half a minute, and with *Native* input size lasts for a few minutes. In our implementation, we deploy python scripts for the high-level control algorithm and machine learning-based power detector and use batch scripts (bash shell) for direct task mapping and migration operations. The tasks and number of parallel threads are randomly chosen and the series of tasks in our test cases are listed in Table IV. The management cycle period is chosen as 2-s while the processor status and IPCM sampling interval are 200 ms. Since we inspect the total execution time of a series of tasks, there will be no idle time for any core. This means once a task is complete on a core(s), this core(s) will be assigned with the next task immediately unless all the tasks in the queue are finished.

In order to make a fair comparison, each run must be launched under the same initial thermal condition. The chip is totally cooled to the initial temperature (about 30 °C) before the next run. And test cases are run many times to minimize the effects of random factors, such as ambient airflow or on-chip data caching. Please note that the cooling efficiency is forced constant all the time during the experiment. Backside liquid circulation is at a constant flow rate, besides, the thermal-electric device which transfers heat from the motherboard downward to the liquid circulation is kept at constant power at 62 W.

As shown in Table V, the proposed technique will not degrade the system performance. Actually, the average execution time of the whole series of tasks is slightly decreased by 1.4%–4.1%. It is interesting that one or two of the slow runs under Linux are considerably longer than the average time, which we are not sure about the reason. However, the execution time by Hot-Trim is quite stable. As mentioned in Section V-B, incoming tasks are launched following an FIFO order assumed by the series (task queue). When conducting the experiment under Linux default mapping, the task execution order is still determined by the FIFO. Essentially, we use a python script to launch the task one after one once there are available cores or previous tasks are done. Task is launched without setting its CPU affinity, hence the core assignment is decided by Linux. In this way, we could maximize the identity of other factors but only leave the mapping decisions to be different when comparing with temperature-based and the proposed algorithm. It is also more realistic that different tasks randomly come in the time axis than launching them all together. Hence, the task order or thread order is not within the scale of this study. The execution time (min, max, and average) listed in Table V pertains to the variation of a single run of the series of tasks.

On the other hand, the Hot-Trim task mapping algorithm reduces the average core temperature by about 1.21 °C–1.31 °C degrees and the temperature is constantly lower for all test cases compared to Linux baseline as shown in Table V. It should be noted that the temperature reductions are all measured from the thermography system. Temperature at the truly identified hot spots reflects the same trend between the two mapping ways. We measured that the maximum temperature at the identified hot spots for different mapping methods is very similar (at around 95 °C). However, the high-temperature duration and temperature spatial distribution vary. The average temperature at the worst hot spot HS3 of each individual core is around 1.5–2 degrees higher than its average core temperature.

We note that the new task mapping method has no obvious effect on suppressing peak temperature in this study. The main reason is that the maximum temperature at the identified hot spots is determined by some heavy tasks, such as *blackscholes*, *bodytrack*, and *fluidanimate* regardless of which core they are

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

TABLE V
LINUX VERSUS HOT-TRIM: PERFORMANCE AND TEMPERATURE

Test Cases	Total Execution Time (seconds)							Avg. Core Temperature (°C)		
	Linux			Hot-Trim			Avg dt (%)	Linux	Hot-Trim	Reduction (dT)
	Min	Max	Avg	Min	Max	Avg				
Case 1	58.16	62.18	61.03	56.15	61.17	59.84	-2.0%	85.98	84.67	-1.31
Case 2	321.6	358.7	341.1	318.6	333.7	327.2	-4.1%	92.10	90.89	-1.21
Case 3	163.3	165.3	164.0	158.3	163.4	161.7	-1.4%	86.39	85.13	-1.26

assigned to as all the cores are homogeneous. As a result, as long as they are executed in the experiments, we will observe the similar maximum temperature regardless of the mapping method used.

B. *Thermal and Reliability Improvement*

This section compares the thermal behavior and VLSI reliabilities regarding EM, NBTI, and HCI among the three mapping methods. In this experiment, under each mapping method, a series of randomly chosen tasks are released one after one with random intervals between releasing two tasks in the timeline to mimic task allocations in real processors. To make a fair comparison and minimize the effects of random factors, the tasks are chosen in a pseudo-random way as well as the release intervals. In this way, all three mapping techniques will deal with identical workloads and identical arrival times of the workloads. To be simple and without losing the generality, we release 30 randomly chosen tasks one after one intermittently. The time interval (Δt) between releasing two consecutive tasks satisfy a uniform distribution $\Delta t \sim U(4, 12)$ s. The minimum and maximum intervals are 4 and 12 s, respectively. This testing scenario is called case 4. To further minimize the effects of random factors, the same series of tasks are executed under every mapping technique many times and each run starts under the same initial thermal condition.

Fig. 11 compares the temperature at the identified hot spot location HS3 with respect to each core under the three mapping techniques when releasing tasks with interval distribution $\Delta t \sim U(4, 12)$. In our case, HS3 is the most stressed one of the three identified primary hot spots. LB, TB, and HT are briefed for Linux baseline, temperature based, and Hot-Trim mapping, respectively. It can be observed from the plots that most of the time the temperature trend under HT is more similar to TB compared with LB. Mean of the temperature curves shown in Fig. 11 at the HS3 for each core are compared in Table VI. Thermal performance under Hot-Trim is obviously better than the Linux baseline across all cores and is 1.15-°C lower on average in this test case. Under temperature-based mapping, the temperature is quite balanced across all cores, which is expected. Though, its average temperature is still higher than Hot-Trim.

Core frequencies under the three mapping policies are shown in Fig. 12. In the experiment case, core frequencies show similar amplitude where the cores operate at around 1–1.2 GHz while under load and gate to near-zero frequency while they are idle. Frequency throttling seems not to show observable differences among the three mapping methods while the cores are under load. We remark that the system DVFS governor remains untouched during the experiments, hence changes in the frequency pattern for all cores are naturally governed by the system DVFS governor, and different mapping methods are treated constantly. It can be observed that the similar core utilization does not necessarily give similar lifetime reliability. A good example is that the averaged

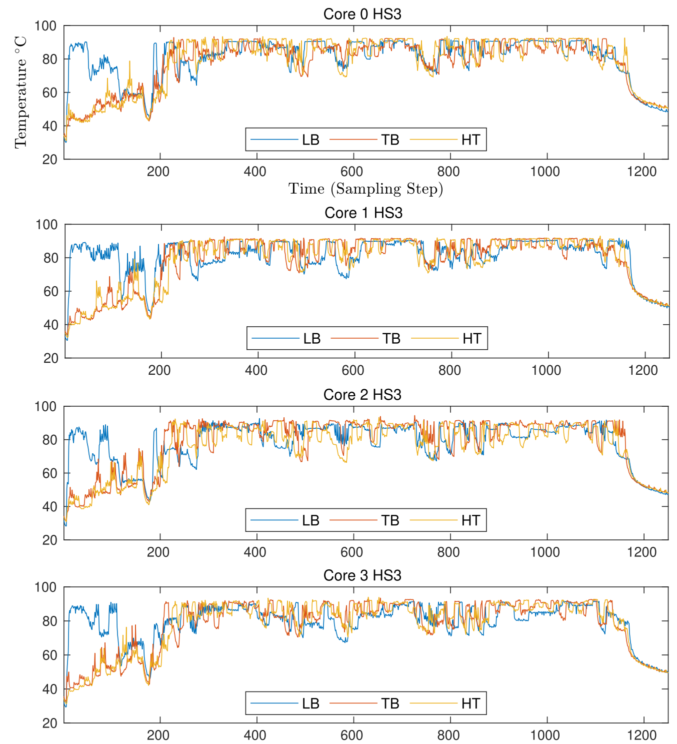

Fig. 11. Temperature at the identified true hot spot location HS3 for each core under three mapping techniques (case 4).

TABLE VI
MEAN OF TEMPERATURE OVER TIME AT THE WORST STRESSED HOT SPOT LOCATION HS3 OF EACH CORE

	Alg.	Core 0	Core 1	Core 2	Core 3	Max	Avg
Case 4	LB	80.81	80.79	77.43	79.18	80.81	79.55
	TB	78.77	79.32	79.11	78.58	79.32	78.94
	HT	79.18	79.43	76.37	78.62	79.43	78.40
Case 5	LB	70.18	65.08	69.64	65.96	70.18	67.72
	TB	66.98	67.52	66.92	67.24	67.52	67.16
	HT	66.76	67.93	64.81	66.71	67.93	66.55

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

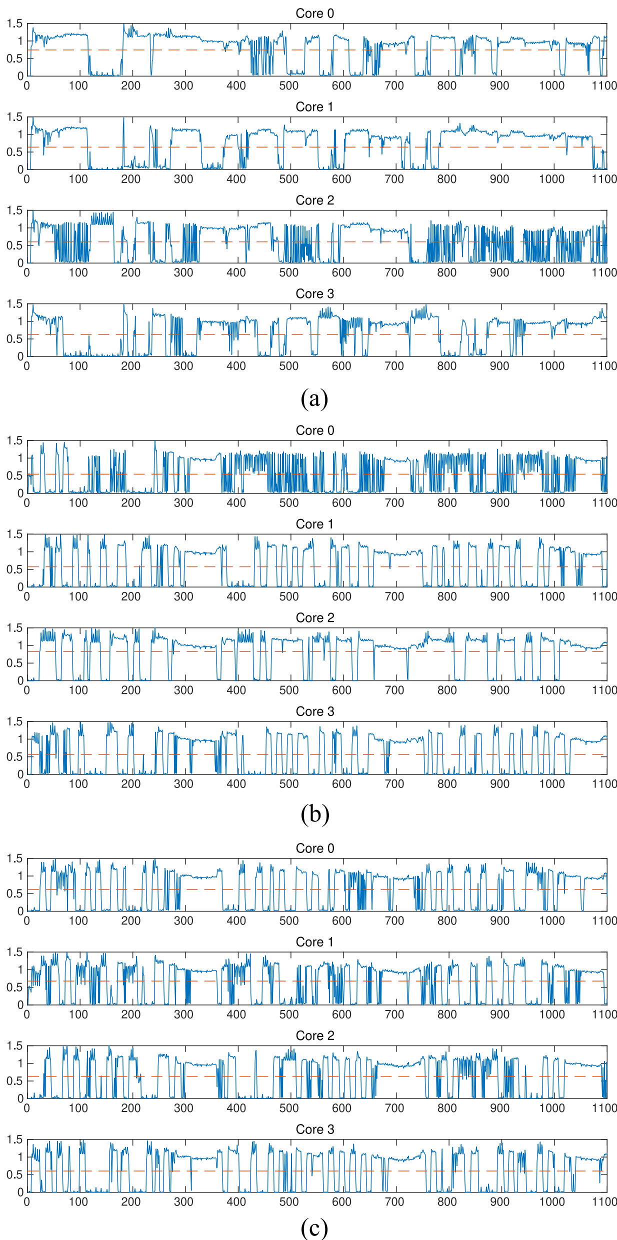

Fig. 12. Core frequencies (GHz) versus time steps when processor under different mapping policies: (a) Linux baseline. (b) Temperature based. (c) Hot-Trim. Red lines are the mean lines of each frequency series (case 4).

frequency of cores 1, 2, and 3 under Linux baseline mapping are very close [about 0.62 GHz, Fig. 12(a)], however, their lifetime reliabilities vary much more, which we will describe in detail later. This is reasonable because cores can have hot spots stressed differently by executing different tasks.

Although temperature does not distinguish much between temperature-based and Hot-Trim mapping, VLSI reliability performances distinguish quite significantly. When implementing analytical models to calculate reliability effects and MTTF, we take advantage of an existing tool called LifeSim [18]. LifeSim is a lifetime reliability simulator that offers a module named reliability management unit (RMU). It calculates MTTF by EM and NBTI effects for many-core systems. For the sake of convenience, we take advantage of the RMU module by feeding our real experiment data, such as core frequency and hot spot temperature to characterize the reliability performance. In the meantime we create another script based on (6) and (7) when calculating MTTF due to HCI, where we treat A_{hci}, $u(V_{dd})$, α, and E_a as simple constants.

As a result, EM-related MTTF is illustrated as Fig. 13(a). Blue, red, and yellow bars stand for Linux baseline, temperature based, and the proposed Hot-Trim mapping, respectively. The y-axis is normalized such that the shortest MTTF among all cores under Linux baseline mapping is ten years. It can be seen that the EM-related MTTF under temperature-based mapping is significantly shorter than Hot-Trim in terms of both average and processor overall. The right-most bar labeled as *Processor* indicates that the MTTF of the entire processor is determined by the minimum MTTF among all the cores. Core 0 is the most stressed under temperature-based mapping whereas core 2 is the most stressed under Linux baseline and Hot-trim. Hot-Trim stresses the cores much more evenly and leads to the longest average and overall MTTF. In detail, Hot-Trim is 30.2% longer than Linux baseline and 29.6% longer than temperature-based mapping in terms of processor overall lifetime, which are very significant.

MTTF due to NBTI is shown in Fig. 13(b). NBTI behavior is quite close between temperature-based and Hot-Trim since the temperature is close between the two mappings, as we know that NBTI is primarily dependent on temperature. The overall MTTF under Hot-Trim is only less than 1% shorter than temperature-based mapping, and 7.0% longer than the Linux baseline. HCI-related MTTF has a similar pattern to the EM-related MTTF, as shown in Fig. 13(c). Specifically, Hot-Trim is 31.1% longer than the Linux baseline and 19.6% longer than the temperature-based technique in terms of overall lifetime.

As for the migration energy overhead, thanks to Intel's Performance Monitor, we measured the entire real processor energy consumption executing the series of tasks as 2106.8, 2129.9, and 2118.5 J under Linux baseline, Temperature-based, and Hot-Trim mapping, respectively. Therefore, the energy variation caused by the algorithm and task migration operations is merely marginal. In our method, the management cycle in the experiment is chosen as 2 s and CPU performance sampling interval is 200 ms. We also implemented 1 s per management cycle and 100 ms per sampling interval and found no measurable difference in the results. The resulting algorithm works quite well for running all the benchmarks. In general, one should reasonably choose the length of the management cycle and sampling interval depending on the user cases when applying the proposed algorithm. Moreover, due to the estimation error of hot spot power densities and the granularity of management, the lifetimes (Figs. 13 and 14) indeed show some imbalances between different cores. Ideally, the VLSI lifetime reliabilities should be perfectly balanced if the hot spot power densities were perfectly estimated.

We present more results through test case 5 to deliver an insight on the change of task release interval (Δt). The only difference in the settings of test case 5 compared to test case 4 is that the distribution of pseudo-random task release intervals is changed to $\Delta t \sim U(10, 20)$ s. This means fewer cores will

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

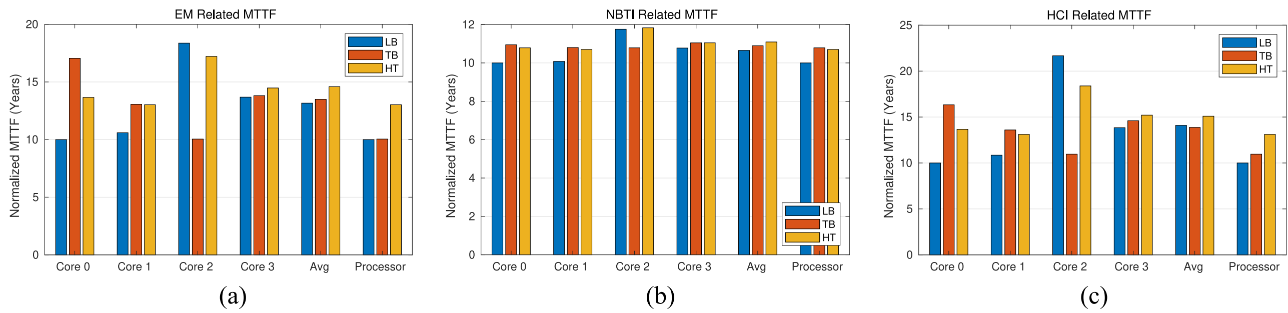

Fig. 13. Normalized MTTF considering (a) EM, (b) NBTI, and (c) HCI reliability effects regarding test case 4 where task release intervals satisfy $\Delta t \sim U(4, 12)$.

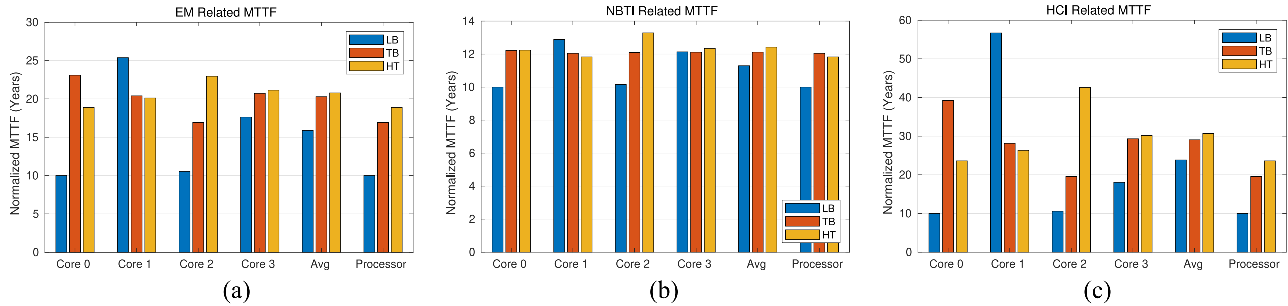

Fig. 14. Normalized MTTF considering (a) EM, (b) NBTI, and (c) HCI reliability effects regarding test case 5 where task release intervals satisfy $\Delta t \sim U(10, 20)$.

be busy at the same time as the arrival of tasks are slower. The mean of temperature traces of the dominant hot spot HS3 for the cores are shown in Table VI. Further, the reliability performances are presented in Fig. 14(a)–(c).

Moreover, proactive dynamic thermal management (PDTM) can be our future research work. The temperature at the identified hot spots can be proactively predicted because the real-time power information at those hot spots can be accurately estimated from IPCMs (Section V-A). With the predicted temperature in advance, we can apply a more comprehensive thermal and reliability management model [35], [36].

VII. Conclusion

In this work, we have proposed a new hot-spot-aware task mapping scheme named *Hot-Trim* to improve the reliability and thermal performance of commercial multicore processors without degrading the system execution performance. Our method is motivated by the observation that the power density hot spots in cores and their reliability in a multicore processor are workload dependent and thus can be exploited to improve the reliability of the system. Experiments on a real Intel Core i7 quad-core processor executing PARSEC-3.0 and SPLASH-2 benchmarks show that the core and hot spot temperature can be even reduced by 1.15 °C–1.31 °C. *Hot-Trim* can improve the chip's EM, NBTI, and HCI related reliability by 30.2%, 7.0%, and 31.1%, respectively, compared to Linux baseline without any performance degradation. Furthermore, it improves EM and HCI-related reliability by 29.6% and 19.6% while further reduces the temperature by half a degree compared to the conventional temperature-based mapping technique, proving that temperature per-core sensing may not lead to the optimal reliability management solution.

References

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May/Jun. 2012.

[2] H. Amrouch, V. M. van Santen, T. Ebi, V. Wenzel, and J. Henkel, "Towards interdependencies of aging mechanisms," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 478–485.

[3] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. Eur. Design Test Conf. (DATE)*, Apr. 2007, pp. 1659–1664.

[4] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. Design Autom. Conf. (DAC)*, 2010, pp. 579–584.

[5] G. Liu, M. Fan, and G. Quan, "Neighbor-aware dynamic thermal management for multi-core platform," in *Proc. Eur. Design Test Conf. (DATE)*, 2012, pp. 187–192.

[6] Z. Liu, S. X.-D. Tan, X. Huang, and H. Wang, "Task migrations for distributed thermal management considering transient effects," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 2, pp. 397–401, Feb. 2015.

[7] S. Pagani, P. D. S. Manoj, A. Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 101–116, Jan. 2020.

[8] S. Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Henkel, and S. X.-D. Tan, "Post-silicon heat-source identification and machine-learning-based thermal modeling using infrared thermal imaging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 694–707, Apr. 2021.

[9] S. Reda, K. Dev, and A. Belouchrani, "Blind identification of thermal models and power sources from thermal measurements," *IEEE Sensors J.*, vol. 18, no. 2, pp. 680–691, Jan. 2018.

[10] J. Zhang, S. Sadiqbatcha, W. Jin, and S. X.-D. Tan, "Accurate power density map estimation for commercial multi-core microprocessors," in *Proc. Design, Autom. Test Europe Conf. (DATE)*, 2020, pp. 1085–1090.

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.

[11] J. Zhang, S. Sadiqbatcha, M. O'Dea, H. Amrouch, and S. X.-D. Tan, "Full-chip power density and thermal map characterization for commercial microprocessors under heat sink cooling," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 5, pp. 1453–1466, May 2022.

[12] H. Khdr, T. Ebi, M. Shafique, H. Amrouch, and J. H. Karlsruhe, "mDTM: Multi-objective dynamic thermal management for on-chip systems," in *Proc. Design, Autom. Test Europe Conf. Exhibition (DATE)*, 2014, pp. 1–6.

[13] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[14] S. Lu, R. Tessier, and W. Burleson, "Reinforcement learning for thermal-aware many-core task allocation," in *Proc. 25th Great Lakes Symp. VLSI*, 2015, pp. 379–384.

[15] A. Iranfar, S. N. Shahsavani, M. Kamal, and A. Afzali-Kusha, "A heuristic machine learning-based algorithm for power and thermal management of heterogeneous MPSoCs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2015, pp. 291–296.

[16] A. Zou, K. Garimella, B. Lee, C. Gill, and X. Zhang, "F-LEMMA: Fast learning-based energy management for multi-/many-core processors," in *Proc. ACM/IEEE Workshop Mach. Learn. CAD*, 2020, pp. 43–48.

[17] J. Zhang, S. Sadiqbatcha, Y. Gao, M. O'Dea, N. Yu, and S. X.-D. Tan, "HAT-DRL: Hotspot-aware task mapping for lifetime improvement of multicore system using deep reinforcement learning," in *Proc. ACM/IEEE Workshop Mach. Learn. CAD*, 2020, pp. 77–82.

[18] R. Rohith, V. Rathore, V. Chaturvedi, A. K. Singh, S. Thambipillai, and S.-K. Lam, "LifeSim: A lifetime reliability simulator for manycore systems," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2018, pp. 375–381.

[19] X. Huang, T. Yu, V. Sukharev, and S. X.-D. Tan, "Physics-based electromigration assessment for power grid networks," in *Proc. 51st Design Autom. Conf.*, 2014, pp. 1–6.

[20] S. X.-D. Tan, M. Tahoori, T. Kim, S. Wang, Z. Sun, and S. Kiamehr, *VLSI Systems Long-Term Reliability—Modeling, Simulation and Optimization*. Cham, Switzerland: Springer, 2019.

[21] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proc. 31st Annu. Int. Symp. Comput. Archit.*, 2004, pp. 276–287.

[22] A. Dasgupta and R. Karri, "Electromigration reliability enhancement via bus activity distribution," in *Proc. 33rd Design Autom. Conf.*, 1996, pp. 353–356.

[23] N. Kimizuka, T. Yamamoto, T. Mogami, K. Yamaguchi, K. Imai, and T. Horiuchi, "The impact of bias temperature instability for direct-tunneling ultra-thin gate oxide on MOSFET scaling," in *Symp. VLSI Technol. Dig. Tech. Papers*, 1999, pp. 73–74.

[24] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, 2005, pp. 520–531.

[25] S. Zafar, B. H. Lee, J. Stathis, A. Callegari, and T. Ning, "A model for negative bias temperature instability (NBTI) in oxide and high /spl kappa/ pFETs 13/spl times/-C6D8C7F5F2," in *Dig. Tech. Papers Symp. VLSI Technol.*, 2004, pp. 208–209.

[26] E. Takeda, C. Y. Yang, and A. Miura-Hamada, *Hot-Carrier Effects in MOS Devices*. San Diego, CA, USA: Academic, 1995.

[27] F. Oboril and M. B. Tahoori, "Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2012, pp. 1–12.

[28] H. Amrouch and J. Henkel, "Lucid infrared thermography of thermally-constrained processors," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2015, pp. 347–352.

[29] K. Dev, A. N. Nowroz, and S. Reda, "Power mapping and modeling of multi-core processors," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Sep. 2013, pp. 39–44.

[30] "Intel performance counter monitor (PCM)." Intel. 2017. [Online]. Available: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[31] "AMD uProf." AMD. 2022. [Online]. Available: https://developer.amd.com/amd-uprof/

[32] S. Sadiqbatcha, J. Zhang, H. Amrouch, and S. X.-D. Tan, "Real-time full-chip thermal tracking: A post-silicon, machine learning perspective," *IEEE Trans. Comput.*, vol. 71, no. 6, pp. 1411–1424, Jun. 2022.

[33] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Dept. Comput. Sci. Eng., Princeton Univ., Princeton, NJ, USA, Jan. 2011.

[34] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 24–36, 1995.

[35] Y. Fu, L. Li, K. Wang, and C. Zhang, "Kalman predictor-based proactive dynamic thermal management for 3-D NoC systems with noisy thermal sensors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 11, pp. 1869–1882, Nov. 2017.

[36] K.-C. J. Chen and Y.-H. Liao, "Adaptive machine learning-based temperature prediction scheme for thermal-aware NoC system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2020, pp. 1–4.

Jinwei Zhang (Student Member, IEEE) received the B.S. degree in electrical and control engineering from the Beijing Institute of Technology, Beijing, China, in 2014, and the M.S. degree in electrical engineering from Washington University in St. Louis, St. Louis, MO, USA, in 2016. He is currently pursuing the Ph.D. degree with the VLSI System and Computation Lab, Department of Electrical and Computer Engineering, University of California at Riverside, Riverside, CA, USA.

His research interests are aligned with electronic design automation, system-level dynamic thermal/reliability management, power modeling and optimization for VLSI, and commercial multicore processors with machine learning techniques.

Sheriff Sadiqbatcha (Student Member, IEEE) received the B.S. degree (most outstanding graduate of the year) in computer engineering from California State University at Bakersfield, Bakersfield, CA, USA, in 2016, the M.S. degree (*magna cum laude*) in electrical engineering from the University of California at Riverside, Riverside, CA, USA, in 2019, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of California at Riverside in 2022.

He is an Associate Instructor with the Department of Electrical and Computer Engineering, University of California at Riverside. His research interests include presilicon reliability (electromigration), and applied machine-learning in the area of electronic design automation, post-silicon thermal, power, and reliability modeling and control.

Sheldon X.-D. Tan (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1992 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from The University of Iowa, Iowa City, IA, USA, in 1999.

He is a Professor with the Department of Electrical Engineering, University of California at Riverside, Riverside, CA, USA, where he is also a Cooperative Faculty Member with the Department of Computer Science and Engineering. He was a Visiting Professor with Kyoto University, Kyoto, Japan, as a JSPS Fellow from December 2017 to January 2018. His research interests include machine and deep learning for VLSI reliability modeling and optimization at circuit and system levels, machine learning for circuit and thermal simulation, thermal modeling, optimization and dynamic thermal management for many-core processors, efficient hardware for machine learning and AI, parallel computing and simulation based on GPU and multicore systems. He has published more than 330 technical papers and has coauthored six books on those areas.

Dr. Tan received the NSF CAREER Award in 2004. He received best paper awards from ICSICT'18, ASICON'17, ICCD'07, and DAC'09. He also received the Honorable Mention Best Paper Award from SMACD'18. He is serving as the TPC Chair for ASPDAC 2021 and the TPC Vice Chair for ASPDAC 2020. He is serving or served as the Editor-in-Chief for *Integration, the VLSI Journal* (Elsevier), the Associate Editor for four journals, including IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS (VLSI) SYSTEMS, *ACM Transactions on Design Automation of Electronic Systems*, *Microelectronics Reliability* (Elsevier), and *Electronics, Microelectronics and Optoelectronics Section* (MDPI).

Authorized licensed use limited to: Univ of Calif Riverside. Downloaded on June 24,2026 at 00:46:27 UTC from IEEE Xplore. Restrictions apply.