Real-Time Full-Chip Thermal Tracking: A Post-Silicon, Machine Learning Perspective

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

Abstract—In this article, we present a novel approach to real-time tracking of full-chip heatmaps for commercial off-the-shelf microprocessors based on machine-learning. The proposed *post-silicon* approach, named *RealMaps*, only uses the existing embedded temperature sensors and workload-independent utilization information, which are available in real-time. Moreover, *RealMaps* does not require any knowledge of the proprietary design details or manufacturing process-specific information of the chip. Consequently, the methods presented in this work can be implemented by either the original chip manufacturer or a third party alike, and is aimed at supplementing, rather than substituting, the temperature data sensed from the existing embedded sensors. The new approach starts with offline acquisition of accurate spatial and temporal heatmaps using an infrared thermal imaging setup while nominal working conditions are maintained on the chip. To build the dynamic thermal model, a temporal-aware long-short-term-memory (LSTM) neutral network is trained with system-level features such as chip frequency, instruction counts, and other high-level performance metrics as inputs. Instead of a pixel-wise heatmap estimation, we perform 2D spatial discrete cosine transformation (DCT) on the heatmaps so that they can be expressed with just a few dominant DCT coefficients. This allows for the model to be built to estimate just the dominant spatial features of the 2D heatmaps, rather than the entire heatmap images, making it significantly more efficient. Experimental results from two commercial chips show that *RealMaps* can estimate the full-chip heatmaps with 0.9°C and 1.2°C root-mean-square-error respectively and take only 0.4ms for each inference which suits well for real-time use. Compared to the state of the art *pre-silicon* approach, *RealMaps* shows similar accuracy, but with much less computational cost.

Index Terms—Post-silicon, thermal modeling, real-time, temperature estimation, processor heatmaps, infrared imaging, machine learning, deep neural networks.

1 Introduction

With the continuing trend of rapid integration and technology scaling, today's high performance processors have become more thermally constrained than ever before. Increase in temperature has been shown to exponentially degrade reliability of semiconductor chips [1], [2], [3], and has consequently become one of the leading concerns in the industry today. Thermal and reliability validation and sign-off is a crucial step in today's physical design flow. Commercial tools exist to ensure sound thermal design from the device level [4] all the way to the system-on-chip (SoC) level [5]. While design time thermal considerations play a crucial role in ensuring reliability and consistent performance, monitoring and managing the processor's temperature while it is in use is equally important. This is especially a challenge for system integrators that produce thin and light mobile devices, laptops, and embedded systems where the space restrictions limit the effectiveness of traditional coolers such as heatsinks and heatpipes. In such cases, software based thermal monitors and controllers can be used along with the external coolers to ensure proper operation. To this end, runtime power and thermal control schemes are being implemented in most, if not all new generations of devices [6], [7]. These control schemes depend on accurate real-time temperature information, ideally of the entire die area of the processor in order to be effective [8], [9].

On-chip temperature sensors alone cannot provide the full-chip temperature information since the number of sensors that can be placed in a chip is limited due to the high design overheads that they incur. It has been shown recently that the number of hotspots on a typical commercial microprocessor far exceeds the number of embedded temperature sensors [10]. Consequently, the thermal and power control algorithms that solely depend on the embedded sensors become oblivious to significant temperature peaks that occur spatially distant from the sensor locations. For example, Fig. 1 shows a significant temperature difference observed between a hotspot and the nearest embedded sensor in an Intel Core i7-8650U processor.

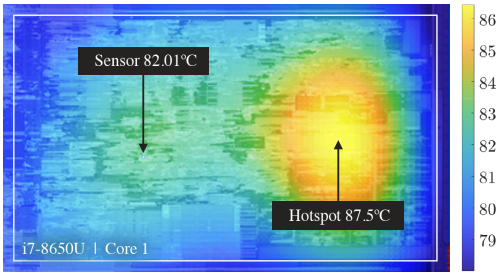


Fig. 1: Hot-spot observed spatially distant from an embedded temperature sensor in an Intel i7-8650U

Adapting smart sensor placement algorithms that aid in spatial temperature interpolation of non-sensor locations can help mitigate this issue [11], [12], [13]. However, these methods require modifications to the chip's design (i.e. adding or relocating temperature sensors) which is not a post-silicon approach that can be applied to off-the-shelf processors.

In this article, we introduce a software-based solution to the problem at hand. Specifically, we present an entirely new data-driven approach, named *RealMaps*, to deriving a light-weight thermal model that is capable of real-time estimation of full-chip spatial heatmaps. The estimated heatmaps from the model can then be used to supplement the temperature readings from the embedded temperature sensors for more effective thermal monitoring and control [14]. To our knowledge, the proposed

- *Sheriff Sadiqbatcha, Jinwei Zhang, and Sheldon X.-D Tan are with the Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521 USA*
- *Hussam Amrouch is with the University of Stuttgart, Suttgart, Germany*

This work is supported in part by NSF grant under No. CCF-1816361, and in part by NSF grants under No. CCF-2007135 and No. OISE-1854276.

approach is the first one that can be implemented on existing commercial multi-core processors for real-time full-chip heatmap estimation. Note that this article presents the complete and comprehensive version of the preliminary work that we published in a conference [15]. The following is a summary of the contributions of this work.

- First, *RealMaps* can be implemented on most, if not all, existing commercial microprocessors and micro-controllers as it only uses the existing temperature sensors and workload independent utilization information. In other words, our strictly post-silicon approach does not require any modifications to the chip's design. Additionally, unlike many existing methods, it requires no proprietary design, floor-plan or process-specific information and therefore can be implemented by both the original chip manufacturer and third-parties, such as system integrators and academic research labs, on future, current, and older generations of microprocessors alike.
- Second, our model is built based on high-level performance monitors, which are supported in most, if not all, commercial microprocessors. High-level performance monitors, unlike low-level performance counters, provide system-level utilization metrics such as the core frequency, instruction counts, cache hit/miss-rates etc rather than functional-unit-wise access rates.
- Third, *RealMaps* uses an advanced infrared (IR) thermography setup that enables lucid heatmaps to be recorded directly from commercial microprocessors while they are under load. This system allows us to build and validate the model using data measured directly from commercial off-the-shelf microprocessors as opposed to using simulation platforms.
- Fourth, to reduce the dimensionality of the model, 2D spatial discrete cosine transformation (DCT) is first performed on the heatmaps so that they can be expressed with just their dominant DCT frequencies. This allows for the model to be built to estimate just the dominant spatial frequencies of the 2D heatmaps, rather than the entire heatmap images, making it significantly more efficient.
- Last but not least, we propose the use of long-short-term-memory (LSTM) neural-networks (NN), which can discern temporal information, to build the model. This popular recurrent-neural-network (RNN) architecture is ideal for extracting features from sequential input data and therefore performs very well in the application at hand where several time-steps of high-level performance metrics are used for each time-step of temperature inference.

Experimental results validated using measured thermal data from two commercial chips (Intel i5-3337U and i7-8650U) show that *RealMaps* can estimate the full-chip heatmaps with $0.9°$C and $1.2°$C root-mean-square (RMS) error with $0.4ms$ of inference time. This makes the proposed approach very desirable for online thermal estimation. Additionally, when compared to the state-of-the-art full-chip heatmap estimation method, *EigenMaps* [13], which requires *pre-silicon* design modifications, our purely post-silicon *RealMaps* shows similar accuracy, but with much less computational cost.

This article is organized as follows. Sec. 2 reviews the existing relevant work. Sec. 3 illustrates the IR thermography setup used in this study. Sec. 4 details the proposed input and output dimensionality reduction technique used to reduce the size of the model. Sec. 5 presents the framework and implementation details of the proposed approach. Sec. 6 presents the experimental results and comparisons with the current state-of-the-art method. Sec. 7 concludes this article.

2 Related Work and Motivation

Hardware performance counters are a collection of special-purpose registers that are now present in most, if not all, commercial microprocessors. These registers can be configured to count low-level performance metrics such as the utilization rates of one or more functional units (FU). Due to the correlation between a FU's utilization rate and its power consumption, it has been shown that FU-wise thermal and power models can be built as functions of low-level performance metrics. Exploiting this correlation, low-level performance metrics and temperature readings from the embedded temperature sensors have been used in the past to predict the future readings from the embedded sensors (i.e. at time t predict $Tsens_{t+1}$) [16]. Predicting the future temperature aids in the development of proactive thermal control schemes as opposed to the existing reactive ones [17], [18]. However, in this study, we attempt to solve a different problem. As previously mentioned, the number of embedded sensors on the chip is very limited due to their high design overheads and they may not always be placed in close proximity to the hot-spots on the chip (Fig. 1). Hence it is imperative to develop methods to monitor the temperature and/or power distributions across the entire chip's surface-area in real-time.

To that end, software-based power-consumption and temperature estimation methods for both high performance and mobile/embedded processors have been developed [19], [20], [21], [22], [23], [24]. These methods offer a software-based solution to runtime FU-wise, or package-wise, power and temperature estimation which otherwise would require a vast number of embedded sensors that incur significant design overheads and are prone to sensing and process-based noise [18]. However, these methods typically rely on manually identifying the low-level performance metrics that are correlated with each FU on the chip. Additionally, for FU-wise estimation, at least one low-level performance metric must be recorded for each FU at all times. However, the number of configurable performance counting registers available in a typical microprocessor is limited, hence simultaneous monitoring of the entire chip is not feasible using this method.

To overcome these challenges, two general strategies have been explored. The first is to estimate the full-chip heatmaps from physics-based thermal models and power related information [25], [26]. These thermal models can be built using the so-called "bottom-up" approaches such as HotSpot [27] based simplified finite difference methods, finite element methods [28], equivalent thermal RC networks [29], and recently proposed behavioral thermal models based on the matrix pencil method [30] and the subspace identification method [31], [32]. However, such methods are typically not suited for real-time use and often require accurate component wise power-traces as inputs, which are not trivial to obtain [33], [34]. Second is to use an interpolation based approach to estimate the full-chip heatmaps from the embedded sensor readings [35]. Since the number of sensors and their placement have a significant impact on the accuracy of the aforementioned interpolation, smart sensor placement algorithms have also been proposed that can be used during design time to find the optimal placement for the given budget of embedded temperature sensors [11], [12], [13], [36], [37]. It has been shown that adapting smart sensor placement algorithms can improve the accuracy of soft-sensing or interpolation based methods that can be used to estimate the temperature of any arbitrary location on the chip [12], [13].

However, the aforementioned methods either require design-time hardware changes (inserting or relocating sensors) or at the very least require detailed knowledge of the chip's floorplan and constants specific to the technology-node which are not disclosed by the original chip manufacturer. An exclusively *post-*

silicon approach to real-time estimation of the spatial temperature distribution across the entire chip area (i.e. at time t, estimate the full-chip spatial heatmap $T(x,y)_t$) remains a challenge for existing commercial microprocessors. Such an approach would aid the original chip manufacturer, as well as third-parties, such as system integrators and academic research labs, in developing more robust thermal, power, and reliability control schemes that can make use of both the real-time temperature data sensed by the existing embedded sensors, as well as the real-time estimation from the thermal model.

On the other hand, recently, machine-learning (especially deep-learning) is gaining much attention due to the breakthrough performance in various cognitive applications such as visual object recognition, object detection, speech recognition, natural language understanding, etc., due to dramatic accuracy improvements in their time-series or sequential modeling capabilities [38]. Machine-learning for electronic design automation (EDA) is also gaining significant traction as it provides new computing and optimization paradigms for many of the challenging design automation problems that are complex in nature. For instance, machine learning methods have been applied to power modeling [39] and design space exploration [40]. Additionally, machine-learning based schemes have recently been explored to build a workload-dependent thermal prediction model [41], where the future steady-state temperature of the chip can be predicted by application characteristics and physical features.

Inspired by the recent breakthrough in deep-learning, in this work we present a machine-learning based framework to *post-silicon* full-chip heatmap estimation for commercial off-the-shelf microprocessors. The proposed method leverages the existing embedded temperature sensors and high-level performance monitors which provide system-level metrics such as core-wise frequency, instruction counts, cache hit/miss-rates, overall energy consumption, etc., providing a comprehensive view of the utilization of the entire microprocessor in real time. Deep learning is used to ascertain the relationship between the system-level utilization behavior of the microprocessor and its thermal behavior. The proposed data-driven modeling strategy is structured such that it can be applied to most, if not all, existing commercial multi-core microprocessors with no knowledge of the proprietary design/floorplan information.

3 Infrared Thermography Setup

With any machine-learning based approach, proper acquisition of the training and testing data is of utmost importance. To develop the proposed thermal model for a given microprocessor, two critical pieces of data must be collected. Specifically, a time-continuous sequence of spatial temperature data and high-level performance metrics of the microprocessor captured in synchronous with each-other. To this end, we have built an IR thermography setup that allows us to synchronously capture heatmaps $(T(x,y)_t)$ and performance metrics $(M(j)_t)$ at a constant frequency $(f = 1/\triangle t)$. Here, x and y are spatial coordinates, t is time, $\triangle t$ is the time-span between two adjacent time-steps, and $1 < j < m$ where m is the total number of metrics supported by the performance monitoring software.

Our IR thermography setup, shown in Fig. 2, is based on the setup proposed in [42]. It features a thermo-electric (Peltier) device mounted on the PCB directly beneath the processor allowing it to be cooled from underneath. This leaves the front side of the processor fully exposed to the IR camera without any interference layer in-between. A programmable DC power supply is used to control the heat-flow through the thermo-electric device so that the operating conditions can be matched to the baseline cooling unit (stock heat-sink) using the calibration method discussed in [42]. Unlike the traditional oil-based front-cooling methods, no de-embedding [34] is required in our setup. It should be noted that this cooling system should only be used for processors that require heat-sinks. Many mobile and embedded processors are designed to be operated without heat-sinks. In such cases, the aforementioned cooling system should not be used during data acquisition.

Detailed description of the IR thermography setup is as follows. The IR camera used in this setup is a FLIR A325sc which supports a maximum imaging resolution of 320×240 pixels (px) with 16-bits of precision per px, and a maximum capturing frequency of 60Hz. The IR sensor is factory calibrated for accuracy across the temperature range of $0°C$ to $328°C$, and resolves the IR spectral range of $7.5\mu m$ to $13\mu m$. A microscope lens is used to achieve the spatial resolution of $50\mu m$ per px. The FLIR A325sc has an internal waveform generator that outputs a square waveform in synchronous with the capture rate of the camera. An I/O module is used to interface the waveform generator to the processor-under-test so that the performance metrics (recorded in the processor) can be synchronized with the thermal data recorded by the IR camera. Mounted on the PCB directly underneath the processor is the thermo-electric-based cooling system which includes a Peltier device powered by a programmable DC power supply. The Peltier device is the primary cooling mechanism, keeping the chip operating at nominal working conditions. Thermal energy flows from the cool-side of the Peltier device to its hot-side, where it must be dissipated in order to ensure proper operation. To this end, an off-the-shelf liquid cooling system is used to cool the hot-side of the Peltier device.

It is crucial to note two details regarding the infrared thermography setup. First, the maximum inference frequency of the proposed model will be ultimately limited by the capturing frequency of the IR camera. In our case, the maximum inference frequency cannot exceed 60Hz. If inference at a higher frequency is desired, then a more advanced IR camera will be required. Second, such as setup does incur a considerable financial cost. Currently, the configuration shown in Fig. 2 costs roughly $15K USD in which the IR camera costs roughly $10K USD. While this is significant, in most circumstances, it can be justified as it is a one time investment. For example, if a system integrator plans to produce 100K mobile/embedded devices using a given commercial processor, then the research and development cost of deriving the runtime thermal model for that processor using the proposed method will be only $0.15 per unit.

4 Model Optimization

In *RealMaps*, the thermal model will be built via training a LSTM-based deep-neural-network (DNN). The DNN will be trained offline using the full-chip heatmaps and high-level performance metrics acquired using our IR thermography setup. In machine-learning terminology, the heatmaps will be our labels (output of the model) while the performance metrics are our features (input of the model). Once the model is trained, the goal is to deploy the model back into the processor in the form of a background application residing in the operating system (OS). This application will, in real-time, feed the performance metrics from the online performance monitor into the model; the model will in-turn periodically output the estimated heatmaps. Since the model is meant for real-time use, it is imperative for it to be as lightweight as possible with minimal overheads in processing time and memory usage.

From the DNN perspective, several techniques, such as weight pruning and quantization, exist for optimizing DNN models. However, in this work we will be utilizing Google's Tensorflow

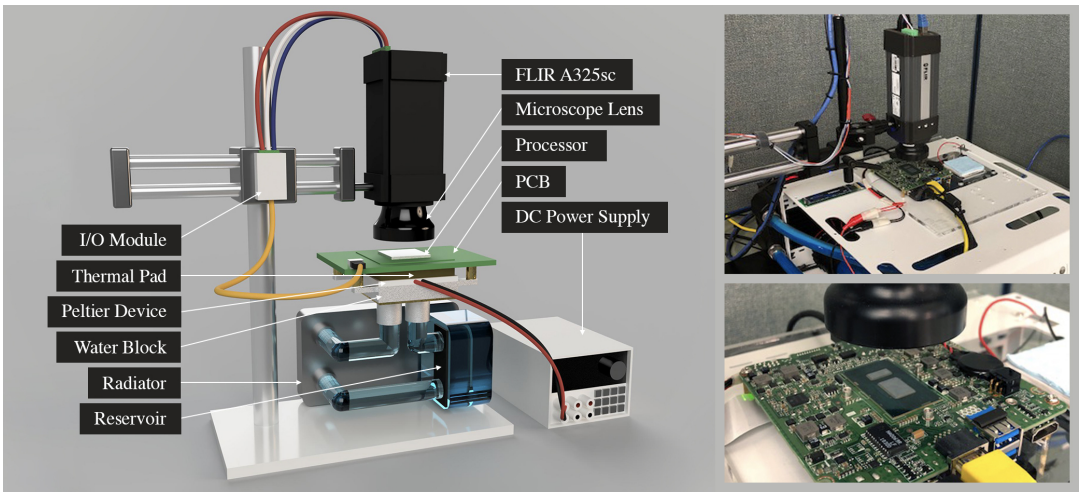


Fig. 2: Our IR thermography setup

(TF) machine-learning library [43] to configure and train our model. The aforementioned weight pruning and quantization methods, along with a number of other optimizations are already performed by the aforementioned library (TF Lite); hence, from this perspective, there's limited margin to optimize our core model any further. We can however optimize the model from the input and output points of view. To this end, in this section we will discuss our approach to output and input dimensionality reduction via dominant spatial frequency extraction and simple cross-correlation analysis.

4.1 Heatmap compression

Heatmaps of the Intel i5-3337U and i7-8650U captured using our IR thermography setup have image resolutions of $177 \times 166px$ and $185 \times 154px$ respectively. This constitutes to a total pixel count of 29382 and 28490 respectively for the two chips. If we were to build DNN models capable of pixel-wise estimation of the full-chip heatmaps (image generation task), then the models for the two chips will need to have output sizes of 29382 and 28490 dimensions respectively. This would not only make the DNN models very large, thus unfit for online inference, but will likely be untrainable due to the large number of trainable parameters that the network will contain. One solution to this problem is to build a model to only estimate the dominant features of the heatmaps rather than the entire heatmap image. The estimated heatmaps can then be reconstructed using the dominant features outputted by the model.

Generally, feature extraction can be carried out using popular dimensionality reduction techniques such as principal component analysis (PCA). Here, compression or approximation is achieved by projecting a data-sample, heatmap $T(x, y)$, onto the subspace spanned by the dominant principal components (PCs) of the available dataset. Such a method would involve first calculating the PCs which form the columns of the change-of-basis matrix. The dominant features in this case would be the coefficients of expansion corresponding to the dominant PCs. However, using this non-standard basis for the application at hand is not recommended since this would require the dominant PCs to be stored in memory. Each PC has the dimensionality equivalent to the pixel-count of the heatmap image (29382 and 28490 single precision floating point values for the two chips respectively). Hence, just storing a few PCs will incur a significant amount of memory overhead. An alternative would be to use a basis with established analytical transformation equations, so that the basis vectors do not need to be stored in memory.

In the case of spatial heatmaps, it has been shown that discrete cosine transformation (DCT) to the spatial frequency domain is an excellent option as the majority of the information from a heatmap can be expressed with just a few low-frequency coefficients of DCT [36]. To this end, we use 2D DCT to convert the measured heatmaps, $T(x, y)$, into spatial frequency-domain [44]. This allows us to extract the dominant low-frequency DCT coefficients of the heatmaps and train our DNN to only estimate these coefficients. Inverse DCT can then be performed at the model's output to recover the estimated heatmaps.

2D DCT is a popular choice for signal and image processing with its "strong energy compaction property" [44]. In most applications, the bulk of the information can be represented by a few low-frequency components of the DCT. A 2D DCT consists of two separate 1D DCT operations, which can be denoted as

$$f_k = \frac{a_0}{\sqrt{N}} + \sqrt{\frac{2}{N}} \sum_{i=1}^{N-1} a_i \cos \frac{(2i+1)k\pi}{2N}, \; 0 \leq k < N, \quad (1)$$

where vector $\{a_i\}$ is the original $(1 \times N)$ data, and $\{f_k\}$ is the result of 1D DCT. A 2D DCT is completed by applying 1D DCT on each column and then on each row of the matrix.

For feature extraction, $1 \leq n \leq x_{max} \times y_{max}$ number of dominant DCT frequencies can be extracted from a spatial heatmap $(T(x, y))$ by transforming $T(x, y)$ into its spatial frequency domain representation $(F(x, y))$ using 2D DCT; then retaining only the first n dominant coefficients in $F(x, y)$.

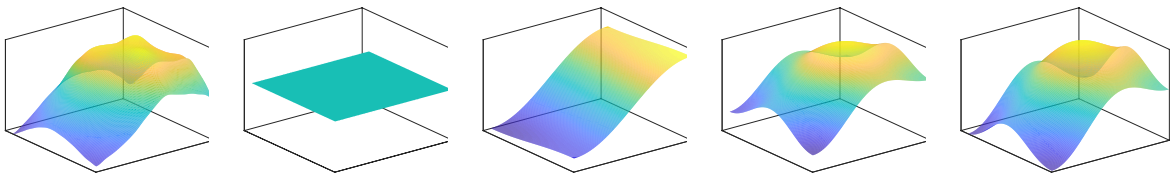
In an image compression scenario, a compressed frequency map $(\mathfrak{F}(x, y))$ is obtained by applying a mask to $F(x, y)$

$$\mathfrak{F}(x, y) = F(x, y)m(x, y), \quad (2)$$

where $m(x, y)$ is a mask map valued 1 at the n most dominant DCT frequency locations and 0 everywhere else. The compressed heatmap $(\mathcal{T}(x, y))$, can then be recovered by carrying out 2D inverse DCT (iDCT) on $\mathfrak{F}(x, y)$. Similar to its forward counterpart (1), 2D iDCT consists of two separate 1D iDCT steps (3) on the rows and columns respectively.

$$a_i = \frac{f_0}{\sqrt{N}} + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} f_k \cos \frac{(2i+1)k\pi}{2N}, 0 \leq i < N. \quad (3)$$

In this case, the higher the value of n, the more $\mathcal{T}(x, y)$ will resemble $T(x, y)$. For example, Fig. 3(a) shows a random heatmap of an Intel i7-8650U. The heatmap compressed using $n = 1$ spatial DCT frequencies is shown in Fig. 3(b). Since only 1



(a) (b) (c) (d) (e)

Fig. 3: (a) A randomly selected uncompressed heatmap of an Intel i7-8650U. (b) Compressed ($n = 1$). (c) Compressed ($n = 3$). (d) Compressed ($n = 6$). (e) Compressed ($n = 10$).

spatial frequency is used, the compressed heatmap only retains the approximate amplitude and no details of the spatial temperature distribution. As n in increased (Fig. 3(c) - Fig. 3(e)), more and more nuanced spatial details are retained in the compression. For a heatmap of size $x_{max} \times y_{max}$, if $n = x_{max} \times y_{max}$, then $\mathcal{T}(x, y) = T(x, y)$.

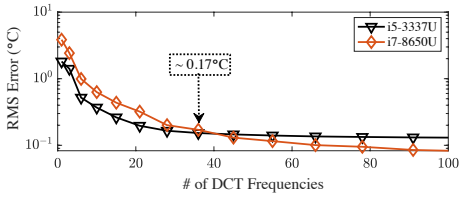


Fig. 4: RMS error between actual heatmaps and heatmaps compressed using varying number of DCT coefficients

In this work, our goal is to identify the minimum value for n that results in minimal loss of spatial information from this compression. In order to determine the minimum number of DCT coefficients that we can use without introducing a significant amount of error, we compress 140,000 heatmaps from each of the two processors with varying number of DCT coefficients. Root-mean-square (RMS) error is then computed between the compressed heatmaps and their uncompressed counterparts. Fig. 4 shows the RMS error in °C as the number of DCT coefficients used in the compression is increased. Based on Fig. 4, we can see that it is sufficient to use only the first 36 most-dominant DCT coefficients, as increasing the number of features further produces marginal benefits. *With this compression, the output of the two models* ($\mathcal{F}$ = **vectorize**(**nonZero**($\mathfrak{F}$)) *only need the dimensionality of 36, instead of 29382 and 28490 respectively. This is a significant reduction to the size of the model, with a compression error of only* $0.17°C$ *RMSE as shown in Fig. 4.*

While we found that 36 DCT coefficients is sufficient for the two chips used in this study, this number may not be true in general for all processor models. Some processors may require fewer, while other may require more. Hence, it is imperative to repeat the above analysis for each processor model to derive the minimum value for n that results in negligible loss due to the compression.

4.2 Performance metrics selection

As previously mentioned, two Intel chips are used in this study. Namely, the Intel i5-3337U (2 cores, 4 threads, released in 2012) representing an older generation and the Intel i7-8650U (4 cores, 8 threads, released in 2017) representing a relatively newer generation of chips from the Intel core family of microprocessors. The primary high-level performance monitoring software supported by Intel is Intel's Performance Counting Monitor (IPCM) [45]. IPCM provides the system-level utilization

TABLE 1: High-level Performance Metrics (Intel PCM)

Package			Core		
Exec	Read	C1res%	Exec_1	L2Miss_1	$\text{c0res\%}_1$
IPC	Write	C2res%	Exec_2	L2Miss_2	$\text{c0res\%}_2$
Freq	INST	C3res%	IPC_1	L3Hit_1	$\text{c1res\%}_1$
AFreq	ACYC	C6res%	IPC_2	L3Hit_2	$\text{c1res\%}_2$
L3Miss	Time	C7res%	Freq_1	L2Hit_1	C3res%
L2Miss	PhysIPC	C8res%	Freq_2	L2Hit_2	C6res%
L3Hit	PhysIPC%	C9res%	Afreq_1	L3MPI_1	C7res%
L2Hit	INSTnom	C10res%	Afreq_2	L3MPi_2	T_{sens}
L3MPI	INSTnom%	Energy(J)	L3Miss_1	L2MPI_1	
L2MPI	C0res%	$_{sens}$	L3Miss_2	L2MPI_2	

The subscript 1 and 2 (i.e. Exec_1 and Exec_2) correspond to hardware threads 1 and 2 within a single core.

metrics that we will be utilizing in this work. For non-Intel chips, the equivalent performance monitors can be used (i.e. AMD uProf [46]).

IPCM provides package and core-wise performance metrics such as energy usage, package and core frequency, instruction counts, cache hit/miss-rates, etc., as well as the sensed temperature from the embedded sensors. Table 1 shows the complete list of IPCM performance metrics from both the package and core-wise domains. There are 30 metrics corresponding to the whole package domain, and 28 metrics for each core. In total, IPCM provides 86 metrics for the dual-core i5-3337U and 142 metrics for the quad-core i7-8650U.

If we were to use the entire IPCM suite as the input to our models, they would have input dimensionalities of 86 and 142 respectively for the two chips. However, although the entire suite of metrics may be useful from a performance monitoring perspective, not all of the metrics may be relevant in modeling the temperature of the chip. Hence, if we can identify the metrics that are irrelevant to the application at hand, eliminating them from the input will aid in further reducing the size of the model. Removing the irrelevant IPCM metrics can be done simply by computing the Pearson's correlation coefficient between a given IPCM metric and each one of the 36 DCT frequencies discussed in Sec. 4.1. If a metric is not correlated with any of the dominant frequencies then it can be eliminated from the input (Algorithm 1). *This trivial approach allows us to reduce the input size from 86 to 58 for the i5-3337U and from 142 to 86 for the i7-8650U.* While this reduction is not as significant as the output reduction in Sec. 4.1, it will nonetheless contribute to the efficiency of the model. Additionally, only utilizing relevant features at the input will make the DNN easier and faster to train.

5 Framework and Implementation

In this section, we will present the framework and implementation specifics of *RealMaps*. We will detail the process of acquiring the necessary data, training and validating the DNN model, and deploying the end model for real-time inference.

Algorithm 1: Relevant IPCM Metric Selection

▷ n = # of DCT Freq, m = # of IPCM metrics, $tmax$ = # of timesteps
input : $\mathbb{F} = [\mathcal{F}_0; ..; \mathcal{F}_{tmax}]$
$\mathbb{M} = [allM_0; ..; allM_{tmax}]$
output: M

$M = [\,]$;
for $i \leftarrow 1$ **to** m **do**
  $maxC = 0$;
  for $j \leftarrow 1$ **to** n **do**
    $currC$ = **PearCorr**($\mathbb{M}[:, i], \mathbb{F}[:, j]$);
    $maxC$ = **max**($currC, maxC$);
  end
  if $maxC \geq 0.5$ **then**
    M = **concatenate**($M, \mathbb{M}[:, i]$);
  else
    pass;
  end
end

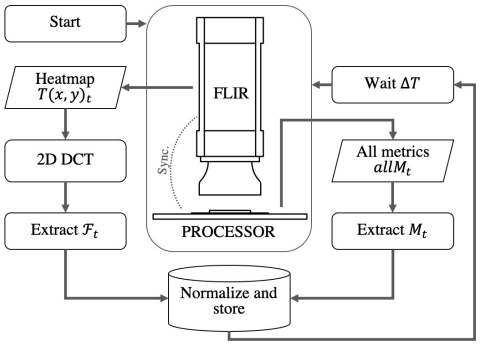


Fig. 5: Data Acquisition Flow

TABLE 2: Phoronix Workloads Executed During Data Acquisition

Processor	System	Memory	Disk
aobench	cyclictest	stream	aio-stress
compress-7zip	phpbench	tinymembench	fio
encode-flac	gimp	t-test1	fs-mark
build-gcc	git	ramspeed	dbench
cachebench	blender	mbw	tiobench

5.1 Data acquisition and normalization

The data acquisition for the proposed approach involves simultaneously recording spatial heatmaps of the processor and performance metrics at a constant capture rate ($f = 1/\triangle t = 60Hz$). At time t, one complete heatmap matrix $T(x, y)_t$ of size $x_{max} \times y_{max}$ is captured, while at the same time IPCM vector $allM_t$ of size $1 \times m$ is recorded. Where $x_{max} \times y_{max} = 177 \times 166$ and $m = 86$ for the i5-3337U and, $x_{max} \times y_{max} = 185 \times 154$ and $m = 142$ for the i7-8650U. The 36 most dominant DCT frequencies (vector $\mathcal{F}_t$), previously identified using the method presented in Sec. 4.1, are extracted from $T(x, y)_t$, while at the same time the relevant IPCM metrics (vector M_t), previously identified using the method presented in Sec. 4.2, are extracted from $allM_t$. Both $\mathcal{F}_t$ and M_t are then normalized to the range of -1 to 1 and saved. After a period of time ($\triangle t$), the next time-step of data is captured in the same manner. This process is repeated until the desired amount of data is acquired. To summarize, the proposed data acquisition flow is illustrated in Fig. 5.

For this study, a total of $t_{max} = 149760$ and $t_{max} = 230400$ time-steps of training data were collected for the i5-3337U and i7-8650U respectively at a capture rate of $60Hz$. This constitutes to a total runtime of 41.6 and 64 minutes respectively. During the initial data collection phase, it is difficult to judge exactly how much data will be needed. However, after the training process, if the model does not produce the desired accuracy then one course of action would be to collect additional data to train the model further. As a rule of thumb, with any machine learning based method, more data will generally lead to a better model. It is however important to denote that increasing the training dataset without increasing the validation dataset can heighten the risk of overfitting. This will result in the model performing well on the training dataset but poorly on the validation dataset and consequently in testing and deployment. In our study, we use 80% of the acquired data for training and 20% for validation. With sufficient validation data, it will be easier to detect and mitigate overfitting during the training process. As we will discuss in the next subsection, this is done by monitoring the learning curve during the training process. If overfitting is detected, then regularization methods can be used.

During the course of data acquisition, the processors were subjected to a variety of workloads. These range from lightweight workloads like idling, to intensive workloads like data compression. Some workloads were primarily compute-intensive tasks while others were memory-intensive. It is important to note that the model itself will be workload independent as it only uses the performance metrics (Sec. 4.2) as the inputs, which contain no information of the workload that the processor is executing. The end goal of the proposed approach is to derive a model that performs reliability regardless of the workload that the processors will be subject to during deployment. However, when the training dataset is being collected, diversity in workloads is nonetheless important due to the extensive use of clock and power gating in modern processor architectures. If a FU is not used when the training data is being collected, it is likely that this FU will remain disabled during the entire time. Hence, it is crucial to utilize all the different sub-systems in the processor during the course of data acquisition so that their thermal behavior can be recorded and consequently be "learned" by the model during the training process. One option to ensure diversity of workloads is to use a benchmark suite that offers a variety of workloads that range in hardware utilization and intensity [47], [48], [49].

For this study, we used the Phoronix test suite [47], which is an open-source benchmark software for Linux that offers an extensive range of workloads. In Phoronix, the workloads are categorized under four domains: processor, system, memory and disk. The processor and system workloads tend to be more compute intensive, while memory and disk workloads tend to be memory intensive. For this study, 5 workloads from each category were randomly selected as shown in Table 2. These workloads were randomly executed in the processor during the course of the data acquisition (Fig. 5).

5.2 Training and testing the LSTM model

As previously mentioned, in this study, we will be employing a LSTM-based DNN to train our online thermal model. A LSTM network is an improved variant of RNNs whose nodes (or neurons) have gated internal states, allowing the network to model problems that require substantial temporal dimensionality. Such a network is ideal for the problem at hand, since the current temperature of a microprocessor (heatmap $T(x, y)_t$) is not just a function of its current utilization (vector M_t), but rather its recent utilization (matrix $[M_{t-s}; ..; M_t]$). In this work we will set $s = 59$, making our estimated heatmap ($\mathcal{T}(x, y)_t$) a function of 60 time-steps

of utilization data ($[M_{t-59}; ..; M_t]$). With the acquisition rate of $60Hz$, 60 time-steps of M represents the processor's utilization for a time-span of 1 second. Given that the thermal time-constant for semiconductor chips is in the order of milliseconds, 1 second of temporal dimensionality at the input should be sufficient. Setting s to a substantially large number will make the model more difficult to train while yielding minimal improvements in accuracy. This is because, during the training process, the weights assigned to the inputs spanning significantly far back in time will be set very close to 0 as their contribution to the current output of the model (current temperature) is minimal.

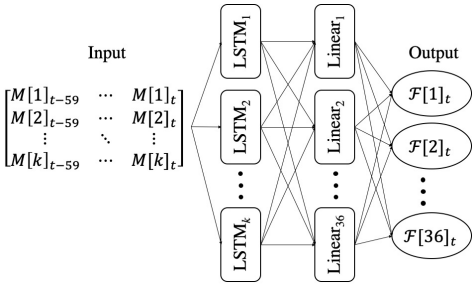


Fig. 6: LSTM Network Configuration

The specific configuration of the LSTM network (# of nodes and # of layers) is not an exact science, especially for a regression problem. Generally, it is recommended to start with a smaller network and increase the size based on its performance. In this work, we will be utilizing the network illustrated in Fig. 6. This is a two layer network with k nodes and 60 time-steps of feedback in the LSTM layer, and 36 nodes in the linear output layer. Through experimentation, we determined that $k = 58$ and $k = 86$ (matching **size**(M)) yielded a good trade-off between network size and inference time for the i5-3337U and i7-8650U respectively.

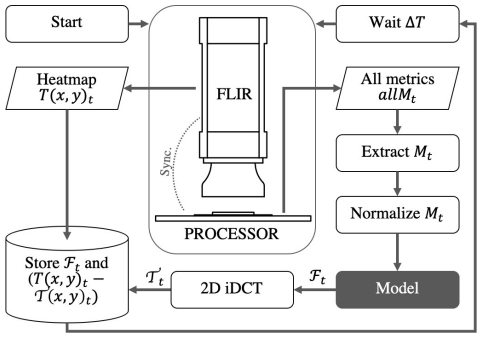


Fig. 7: Testing Flow

After the training data was acquired using the method outlined in Sec. 5.1, the aforementioned LSTM network was trained for a total of 150 EPOCHS with 80% of the data used for training and 20% used for validation. Training was carried out on a server with two Intel 22-core E5-2699 CPUs, and 320GB of memory. The total time elapsed for training was approximately 81 and 112 hours for the two models respectively. As previously mentioned, for each time-step t, 60 time-steps of IPCM metrics $[M_{t-s}; ..; M_t]$ were used as the features (input) while 1 time-step of the 36 most dominant DCT frequencies $\mathcal{F}_t$ extracted from $\mathcal{T}(x, y)_t$ was used as the label (output).

During the training process, it is essential to monitor the learning curve. If validation loss diverges significantly from training loss, this typically indicates overfitting. The learning curves for the two chips are shown in Fig. 8. In our case, overfitting was not found to be an issue. However, this will not always be true as overfitting is a common problem encountered during training. If overfitting is detected, then regularization techniques such as the popular Dropout [50] method can be used to mitigate it.

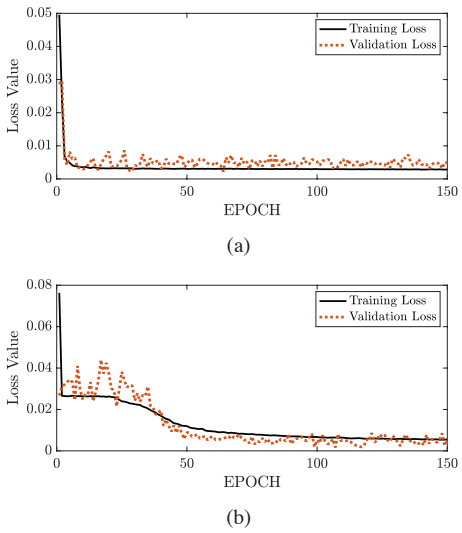


Fig. 8: Learning curves: (a) i5-3337U (b) i7-8650U.

Once the model is trained, it is crucial to test it throughly with new data that was not used for training. As illustrated in Fig. 7, in this study, testing was done by capturing additional thermal data using the IR thermography setup and comparing the measured heatmaps ($T(x, y)$) with the estimated heatmaps ($\mathcal{T}(x, y)$) produced by the model. The error maps ($T - \mathcal{T}$) are saved in order to evaluate the accuracy of the model. For testing, it is recommended to execute a variety of workloads, ideally different than the ones used previously during the acquisition of the training dataset, in order to test both the accuracy and the generality of the model. In this study, the Phoronix test suite [47] was once again used for this purpose. Other benchmark suites such as PARSEC [48], and SPEC [49] or any random collection of workloads that vary in hardware utilization and intensity will suffice as well. The randomly selected Phoronix workloads that were executed during the testing process are: cloverleaf, compilebench, cpp-perf-bench, himeno, pgbench, phpbench, bork, byte, node-octane, opt carrot, osbench, pyperformance, opencv-bench, pybench, sqlite-speedtest, ozone, postmark. This testing process allows us to calculate the error between the estimated heatmaps and the true measured heatmaps, while at the same time, measure the processing and memory overheads of the model. These results for both of the chips will be presented in the next section.

5.3 Deployment

Once the model has been trained and validated as shown in Sec. 5.2, it can be deployed in the processor as a OS-resident background application (software thermal monitor). Online inference can be achieved by directing the performance metrics to the thermal monitor, which in-turn outputs the estimated heatmap ($\mathcal{T}(x, y)_t$). This estimated heatmap can then be directed to a thermal/power controller or any other OS-resident application as desired. After a period of $\triangle t$, this process can be repeated again (Fig. 9). Note, it is important to set $\triangle t$ to be equivalent to the capture rate of the training datasets. In our case, it would be $\triangle t = 1/60sec$ since the training data was captured at the rate of $60Hz$. In the next section, we will present the experimental results from implementing the proposed framework on two test chips and analyze the model's performance both in terms of its estimation accuracy and its overheads compared to the current state-of-the-art approach.

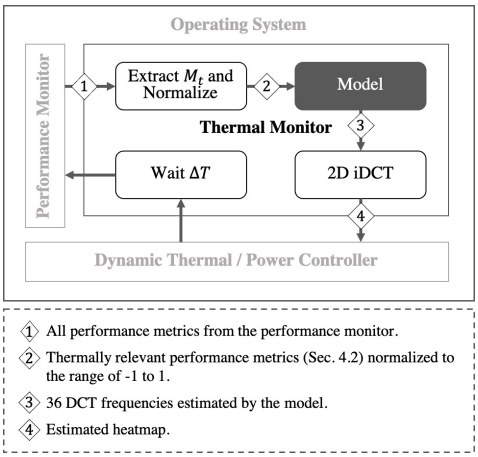


Fig. 9: Model Deployment

TABLE 3: Error stats - Realmaps

	RMS(E)	Mean(E)	Med(E)	Max(E)	Stdev(E)
i5-3337U	0.87°C	0.75°C	0.69°C	10.31°C	0.56°C
i7-8650U	1.24°C	0.86°C	0.70°C	9.74°C	0.75°C

6 Experimental Results and Comparisons

6.1 Experimental results

As outlined in Sec. 5.1 and Sec. 5.2, an extensive set of data were collected from both the i5-3337U and i7-8650U which were then used to train the respective RNN-based thermal models. Once the loss function saturates at a sufficiently low value, the two models were retrieved and put under the testing process illustrated in Fig. 7. As previously mentioned, the testing phase involves utilizing the model to estimate the processor's spatial heatmaps ($\mathcal{T}(x,y)$) while at the same time, the real heatmaps ($T(x,y)$) are captured using the IR thermography setup. At each time-step, the error-maps ($T-\mathcal{T}$) are stored in order to compute the overall accuracy of the model.

Extensive testing conducted on the two chips show that the models perform exceptionally well. The results from the testing process are presented in Fig. 10, Fig. 11, and Table. 3. The estimated 36 DCT coefficients $\mathcal{F}$ follow the trends of the measured data with marginal error. The estimated DCT coefficients for both chips are shown in Fig. 10, plotted along with their measured counterparts from the testing process. For better visualization, we can randomly select a measured heatmap ($T(x,y)_t$) and compare it with the estimated heatmap ($\mathcal{T}(x,y)_t$) from the same testing time-step t. Fig. 11 shows a measured heatmap of the two chips alongside the estimated heatmaps generated by the models. The error-map ($T-\mathcal{T}$) is also shown. The heatmaps shown in Fig. 11 are from the randomly selected time-step $t=15059$ and $t=30073$ from the testing process for the two chips respectively.

In order to formally compute the overall accuracy of the model from the data acquired through the testing phase, we first assemble the error vector given in (4).

$$E = vectorize([T_1 - \mathcal{T}_1, .., T_{tmax} - \mathcal{T}_{tmax}]) \quad (4)$$

Where matrices T and $\mathcal{T}$ are the measured and estimated heatmaps respectively, and $tmax$ is the final testing time-step. The total length of E is 8.8×10^8 and 1.3×10^9 elements for the i5-3337U and i7-8650U respectively. This error vector (E) captures all of the pixel to pixel errors between the measured heatmaps and the estimated heatmaps throughout the span of the entire testing process. Once E has been assembled, the error statistics shown in Table 3 can be calculated.

As shown in Table 3, the models yielded a root-mean-square error of $0.87°$C and $1.24°$C, max error of $10.31°$C and $9.74°$C and a mean error of $0.75°$C and $0.86°$C with a standard deviation of $0.56°$C and $0.75°$C for the i5-3337U and i7-8650U respectively. We believe that this is sufficient for full chip heatmap estimation, especially when considering the fact that the embedded temperature sensors are rated to have an error of $\pm5°$C [51]. As previously mentioned, the heatmaps estimated by the model are to be used to supplement the temperature data sensed by the embedded temperature sensors, rather than being used as a substitute. For example, the readings from the embedded temperature sensors provide more accurate temperature data but only of a few pre-selected locations of the chip, whereas the proposed thermal model will offer full-chip temperature information albeit at lower accuracy. Hence, both the sensors and proposed model can be used together by the dynamic thermal and power controller in order to make a well informed regulation decision. In addition, the accuracy of the model is comparable to the existing pre-silicon techniques which require specialized sensor placement algorithms to be adapted during design time and often require more sensors than the quota allocated for typical microprocessors [24], [36], [52], [53]. As we will show in the next subsection, this model is also more lightweight in terms of the computation and memory overheads when compared to the current state-of-the-art. The computation time of our model is, on average, 0.41 milliseconds per inference for both chips and its memory overhead, primarily incurred in storing the network weights, is 266Kb and 557Kb respectively for the i5-3337U and i7-3650U respectively. This makes the proposed full-chip heatmap estimation technique not only practical, but also highly desirable for online temperature estimation.

One caveat that should be noted, however, is the existence of variations between processor samples that stem from the manufacturing process. The model that is derived using the proposed framework must be robust against such variations. In this study, only one sample of the i5-3337U and i7-8650U were used to collect the training dataset. However, in reality, it is recommended to use multiple samples of the given chip for both the acquisition of the training datasets, as well as for testing the end model. This aids in increasing the robustness of the model to such statistical variations.

6.2 Comparisons with the state-of-the-art pre-silicon approach

As previously mentioned, to our knowledge, the proposed framework is the first exclusively *post-silicon* approach to achieve real-time full-chip spatial heatmap estimation for commercial off-the-shelf microprocessors. However, as discussed in Sec. 2, pre-silicon methods based on smart embedded temperature sensor placement algorithms have been presented in the past [13], [36]. The current state-of-the-art pre-silicon method known as "*Eigenmaps*" was established by Ranieri *et. al.* [13]. In this subsection, we present the implementation of [13] for the two Intel chips used in this study (i5-3337U and i7-8650U) and compare the heatmap estimation accuracy as well as the overheads with that of the proposed post-silicon machine-learning based approach (*Realmaps*).

In summary, the approach in [13] is based on identifying the ideal basis (matrix Φ) for the vectorized spatial heatmaps of the given chip. Sensor locations are determined by calculating the correlation between all the rows of Φ and determining s least correlated spatial locations for sensor placement. Once the sensors are placed, the temperature readings from the sensors can be used to approximate the coefficients of expansion over Φ.

More specifically, let $v[i]$, where $0<i<N=x_{max}\times y_{max}$, be the 1D vectorized form of the 2D heatmap $T(x,y)$, where

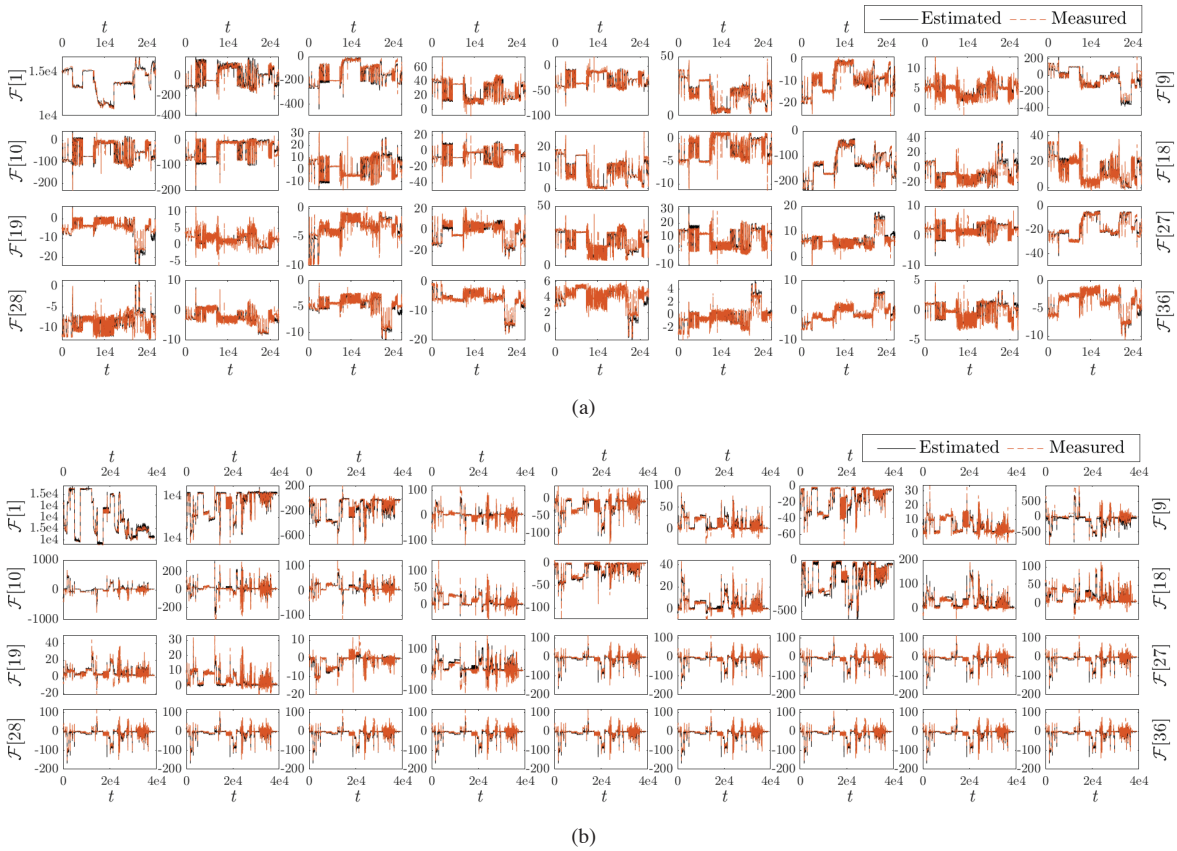


Fig. 10: Estimated vs measured $\mathcal{F}[1]$ to $\mathcal{F}[36]$ (a) i5-3337U (b) i7-8650U.

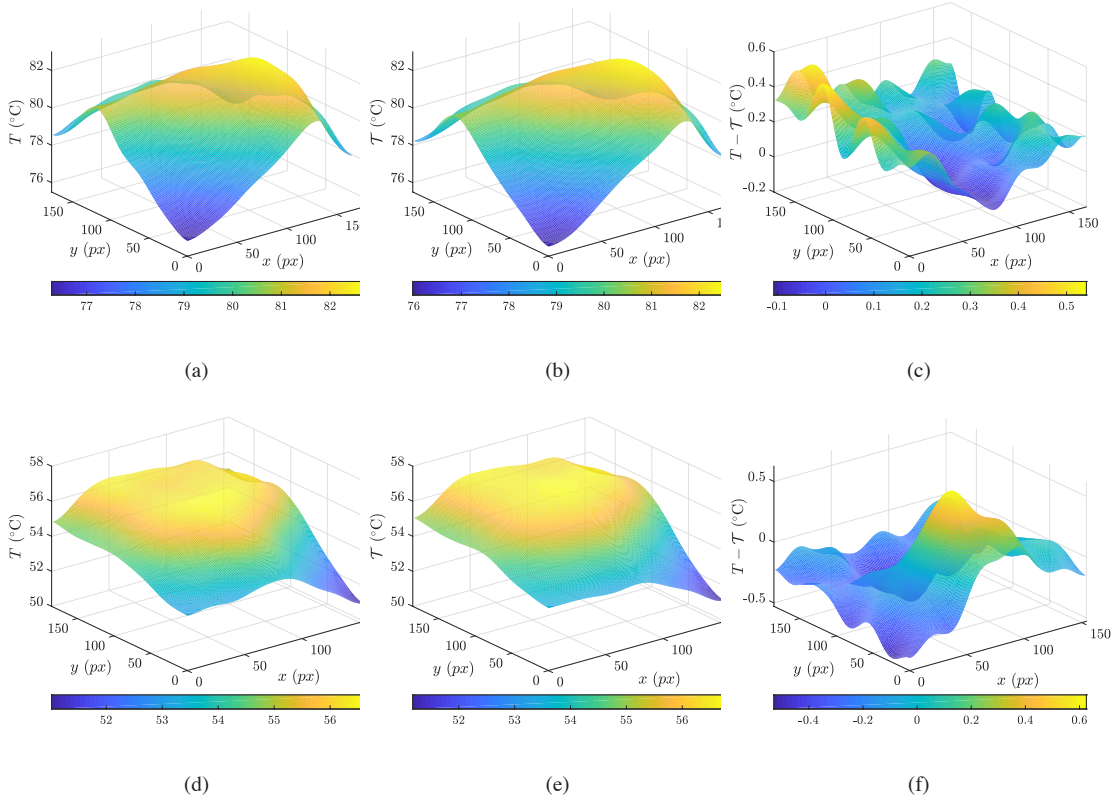


Fig. 11: (a) Measured $T(x, y)$ i5-3337U. (b) Estimated $\mathcal{T}(x, y)$ i5-3337U. (c) Error $T - \mathcal{T}$ i5-3337U. (d) Measured $T(x, y)$ i7-8650U (b) Estimated $\mathcal{T}(x, y)$ i7-8650U (c) Error $T - \mathcal{T}$ i7-8650U.

$0 < x < x_{max}$, $0 < y < y_{max}$, and x_{max} and y_{max} are the horizontal and vertical pixel counts of $T(x, y)$ respectively. This vectorization is done simply by vertically stacking the columns of $T(x, y)$ such that the 1D index i for v corresponds to the 2D index x, y for T according to (5)

$$v[i] = T\left[i \bmod y_{max},\ \left\lfloor \frac{i}{x_{max}} \right\rfloor\right] \tag{5}$$

In general, the vector v can be represented using a basis Φ as per (6).

$$v[i] = \sum_{j=1}^{N} \Phi[i, j]\alpha[j] \tag{6}$$

Here, vector $\alpha[j]$ contains the coefficients of expansion over basis Φ. With vectorized heatmap v expressed as (6), an approximate or compressed vectorized heatmap $\hat{v}$ can be described as a linear combination of the first K columns of Φ and the corresponding K elements of α as shown in (7). In other words, this process is a projection of the spatial heatmap onto the linear subspace spanned by the first K columns of basis Φ.

$$\hat{v} = \Phi[:, 0 : K]\alpha[0 : K] = \Phi_K \alpha_K \tag{7}$$

The optimal subspace Φ_K is the one that introduces the smallest error between v and $\hat{v}$. Finding this optimal subspace Φ_K is a classical problem that is better known as Principal Component Analysis (PCA). As per PCA, the ideal subspace Φ_K is the matrix whose columns are made up of the first K principal components (PCs) of matrix $V = [v_1, .., v_{tmax}]$. Ranieri *et. al.* aptly named these PCs as "*Eigenmaps*" as the analytical solution to computing the PCs involves calculating the Eigenvectors of the covariance matrix of V.

In [13], being a pre-silicon approach, V is derived by simulating the chip's layout using a thermal simulator. At each simulation time-step t, the simulated heatmap $T(x, y)_t$ is vectorized into v_t and stacked into column $V[:, t] = v_t$. This process is repeated until the final simulation time-step t_{max}. In this comparison, we will be using the measured heatmaps of our two chips (our entire training dataset from Sec. 5.1) as a substitute to the simulated heatmaps used in [13].

Similar to the DCT basis discussed in Sec. 4.1, here the higher the value of K, the better $\hat{v}$ will resemble v. However, since Φ_K is the ideal basis for the given problem, far fewer coefficients will be needed compared to the DCT basis. For example, Fig. 12 shows the RMSE computed between the measured heatmaps and the compressed counterparts using varying number (K) of columns in Φ. Comparing this with the same analysis done previously for the DCT basis (Fig. 4), it is clear that Φ_K is indeed a superior basis for the problem. However, the disadvantage of using Φ_K for real-time applications is that it has to be held in memory, which incurs a considerable memory overhead. Note, each columns of Φ_K contains $N = x_{max} \times y_{max}$ single-precision floating point values.

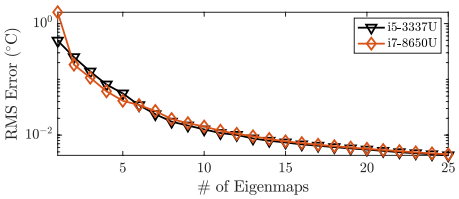


Fig. 12: RMS error between actual heatmaps and heatmaps compressed using varying number of Eigenmaps.

After Φ_K was calculated for the i5-3337U and the i7-8650U, the greedy sensor placement algorithm from [13] was implemented for the two chips to determine the optimal sensor locations given the allocation of s number of temperature sensors. Normally, after the sensors are placed in the design, and the chip is manufactured, the temperature readings from the embedded sensors (vector $v_s = [T_{sens\#1}, .., T_{sens\#s}]$) can be used to approximate the estimated vectorized spatial heatmap $\tilde{v}$ using (8).

$$\tilde{v} = \Phi_K(\tilde{\Phi}_K * \tilde{\Phi}_K)^{-1} * \tilde{\Phi}_K * v_s = C * v_s \tag{8}$$

Here $\tilde{\Phi}_K$ is the matrix made up of the first s columns of Φ_K and the rows corresponding to the spatial coordinates of the selected sensor locations.

For our implementation of *EigenMaps*, we cannot physically embed the sensors as this would require the two chips re-manufactured with the sensors in place. Instead, sensor temperatures are **not sampled** from physical sensors but rather sampled from the measured heatmaps from our thermal imaging system. This is done by simply reading the temperature of the spatial locations where the sensor would have been located. For this comparison, RMSE between the measured heatmaps and the heatmaps estimated using the framework in [13] was calculated using various number of artificially embedded sensors, whose locations are determined by *EigenMaps'* sensor placement algorithm. The results for the two chips is shown in Fig. 13. The results show that the error between the estimated and measured heatmaps generally tend to decrease as the number of allocated sensors is increased. In reality, the number of sensors allocated for a processor typically depends on its core count. This is often equal to the number of cores + 1. For example, the dual-core i5-3337U and the quad-core i7-8650U have 3 and 5 sensors respectively. Accordingly, we will consider $s = 3$ sensors, and $s = 5$ sensors for the two chips in this comparison. The accuracy results for Eigenmaps are presented in Table 4 for the two chips. As the results show, the estimation accuracy of Eigenmaps is comparable to what was achieved with Realmaps (Table 3), especially considering that Eigenmaps requires pre-silicon design considerations, where as Realmaps is an exclusively post silicon framework.

TABLE 4: Error stats - Eigenmaps

	RMS(E)	Mean(E)	Med(E)	Max(E)	Stdev(E)
i5-3337U	0.86°C	0.57°C	0.34°C	9.02°C	0.65°C
i7-8650U	0.94°C	0.57°C	0.26°C	12.52°C	0.74°C

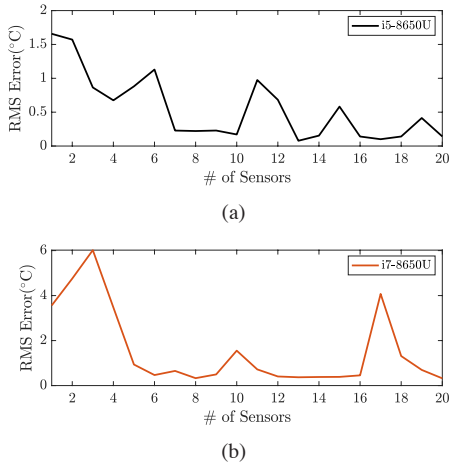


Fig. 13: RMS error between measured heatmaps and heatmaps estimated using [13] as a function of the number of embedded temperature sensors. (a) i5-3337U (b) i7-8650U.

In terms of overheads, deploying the method in [13] for real-time inference would require the expression in (8) to be calculated for each inference. This is computationally very expensive and is therefore not suited for online use. Alternatively, the matrix C in (8) can be pre-calculated and stored in memory. This way, each inference is simply a matrix-vector-multiplication operation. While this is the more suitable option, it does require the entire matrix C to be stored in memory. Note, matrix C is an $N \times N$ matrix whose exact size is 863301924 and 811680100 single-precision floating point elements for the two chips respectively. This translates to a memory overhead of 3.45GB and 3.25GB respectively, which is quite expensive. This can however be remedied by considering lower resolution heatmaps as done in [13].

We remark that the comparison against *EigenMaps* [13] is not an apples-to-apples comparison as *EigenMaps* is a *pre-silicon* approach that requires the placement of embedded sensors at specific locations during design time to achieve the reported accuracy. However, the above comparison does show that the proposed data-driven *RealMaps* framework can yield very similar results in terms of accuracy, with a substantially lower computational overhead. Additionally, the *post-silicon* nature of *RealMaps* makes it feasible for existing commercial off-the-shelf processors. Moreover, it can be used by third parties who do not have control over and, in most cases, have no knowledge of the proprietary design details of the chip. This includes system integrators interested in developing ultra compact mobile devices that benefit from innovative thermal monitoring and management software, and academic research labs that can use the full chip temperature estimation to develop advanced thermal control schemes.

7 Conclusion

In this article, we have proposed a machine learning based framework to real-time estimation of full-chip heatmaps for commercial microprocessors. The proposed approach, named *RealMaps*, only uses the existing embedded temperature sensors and system level utilization information, which are available in real-time. Moreover, it is structured to not require any knowledge of the proprietary design details or manufacturing process-specific information of the commercial processors. Consequently, the methods presented in this work can be implemented by either the original chip manufacturer or a third party alike. In this new approach, we start with accurate spatial and temporal heatmaps measured from an advanced infrared thermal imaging system. To build the transient thermal model, we utilize temporal-aware long-short-term-memory (LSTM) networks with system-level variables such as chip frequency, voltage, and instruction counts as inputs. Instead of a pixel-wise heatmap estimation, we use 2D spatial discrete cosine transformation (DCT) on the heatmaps so that they can be expressed with just a few dominant DCT coefficients. Our study shows that only 36 DCT coefficients are required to maintain sufficient accuracy. Experimental results show that *RealMaps* can estimate the transient heatmaps with $0.9°$C and $1.2°$C RMSE with minimal overheads for the two commercial chips tested in this study. Compared to the state-of-the-art *pre-silicon* method, the proposed approach shows similar accuracy, but with much less computational cost.

References

[1] "Critical Reliability Challenges for The International Technology Roadmap for Semiconductors (ITRS)," 2003. In International Sematech Technology Transfer Document 03024377A-TR, 2003.

[2] S. Sadiqbatcha, Z. Sun, and S. X. . Tan, "Accelerating electromigration aging: Fast failure detection for nanometer ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 4, pp. 885–894, 2020.

[3] S. X.-D. Tan, M. Tahoori, T. Kim, S. Wang, Z. Sun, and S. Kiamehr, *VLSI Systems Long-Term Reliability – Modeling, Simulation and Optimization*. Springer Publishing, 2019.

[4] "Ansys totem." https://www.ansys.com/products/semiconductors/ansys-totem.

[5] "Ansys redhawk." https://www.ansys.com/products/semiconductors/ansys-redhawk.

[6] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *Micro, IEEE*, vol. 32, pp. 122–134, May 2012.

[7] M. Taylor, "A landscape of the new dark silicon design regime," *IEEE/ACM International Symposium on Microarchitecture*, vol. 33, pp. 8–19, October 2013.

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

[9] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Comput. Surv.*, vol. 44, pp. 13:1–13:42, jun 2012.

[10] S. Sadiqbatcha, H. Zhao, H. Amrouch, J. Henkel, and S. X.-D. Tan, "Hot spot identification and system parameterized thermal modeling for multi-core processors through infrared thermal imaging," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019.

[11] R. Cochran and S. Reda, "Spectral techniques for high-resolution thermal characterization with limited sensor data," in *Proc. Design Automation Conf. (DAC)*, pp. 478–483, 2009.

[12] S. Reda, R. Cochran, and A. N. Nowroz, "Improved thermal tracking for processors using hard and soft sensor allocation techniques," *IEEE Transactions on Computers*, vol. 60, pp. 841–851, June 2011.

[13] J. Ranieri, A. Vincenzi, A. Chebira, D. Atienza, and M. Vetterli, "Eigenmaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 636–641, ACM, 2012.

[14] 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 *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, MLCAD '20, (New York, NY, USA), p. 77–82, Association for Computing Machinery, 2020.

[15] S. Sadiqbatcha, Y. Zhao, J. Zhang, H. Amrouch, J. Henkel, and S. X. D. Tan, "Machine learning based online full-chip heatmap estimation," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 229–234, 2020.

[16] R. Cochran and S. Reda, "Consistent runtime thermal prediction and control through workload phase detection," in *Proc. Design Automation Conf. (DAC)*, pp. 62–67, 2010.

[17] A. Bartolini, R. Diversi, D. Cesarini, and F. Beneventi, "Self-aware thermal management for high-performance computing processors," *IEEE Design Test*, vol. 35, pp. 28–35, Oct 2018.

[18] R. Diversi, A. Tilli, A. Bartolini, F. Beneventi, and L. Benini, "Bias-compensated least squares identification of distributed thermal models for many-core systems-on-chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, pp. 2663–2676, Sep. 2014.

[19] M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz, "Practical power consumption estimation for real life hpc applications," *Future Generation Computer Systems*, vol. 29, pp. 208–217, 2013.

[20] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded cpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, pp. 106–119, Jan 2017.

[21] F. Pittino, F. Beneventi, A. Bartolini, and L. Benini, "A scalable framework for online power modelling of high-performance computing nodes in production," *2018 International Conference on High Performance Computing Simulation (HPCS)*, pp. 300–307, July 2018.

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

[23] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware dvfs," *IEEE Transactions on Computers*, vol. 59, pp. 127–133, Jan 2010.

[24] H. Wang, S. X.-D. Tan, S. Swarup, and X. Liu, "A power-driven thermal sensor placement algorithm for dynamic thermal management," in *Proc. Design, Automation and Test In Europe Conf. (DATE)*, pp. 1215–1220, March 2013.

[25] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang, "ISAC: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 1, pp. 86–99, 2007.

[26] H. Wang, S. X.-D. Tan, G. Liao, R. Quintanilla, and A. Gupta, "Full-chip runtime error-tolerant thermal estimation and prediction for practical thermal management," in *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2011.

[27] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 501–513, May 2006.

[28] S. P. Gurrum, Y. K. Joshi, W. P. King, K. Ramakrishna, and M. Gall, "A compact approach to on-chip interconnect heat conduction modeling using the finite element method," *Journal of Electronic Packaging*, vol. 130, pp. 031001.1–031001.8, September 2008.

[29] Y. C. Gerstenmaier and G. Wachutka, "Rigorous model and network for transient thermal problems," *Microelectronics Journal*, vol. 33, pp. 719–725, September 2002.

[30] D. Li, S. X.-D. Tan, E. H. Pacheco, and M. Tirumala, "Parameterized architecture-level dynamic thermal models for multicore microprocessors," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 15, no. 2, pp. 1–22, 2010.

[31] T. Eguia, S. X.-D. Tan, R. Shen, D. Li, E. H. Pacheco, M. Tirumala, and L. Wang, "General parameterized thermal modeling for high-performance microprocessor design," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 2011.

[32] Z. Liu, S. X.-D. Tan, H. Wang, Y. Hua, and A. Gupta, "Compact thermal modeling for packaged microprocessor design with practical power maps," *Integration, the VLSI Journal*, vol. 47, January 2014. in press, online access: http://www.sciencedirect.com/science/article/pii/S0167926013000412.

[33] W. Wu, L. Jin, J. Yang, P. Liu, and S. X.-D. Tan, "Efficient power modeling and software thermal sensing for runtime temperature monitoring," *ACM Trans. on Design Automation of Electronics Systems*, vol. 12, no. 3, pp. 1–29, 2007.

[34] K. Dev, A. N. Nowroz, and S. Reda, "Power mapping and modeling of multi-core processors," in *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 39–44, Sept 2013.

[35] F. Beneventi, A. Bartolini, P. Vivet, and L. Benini, "Thermal analysis and interpolation techniques for a logic + wideio stacked dram test chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 623–636, April 2016.

[36] A. Nowroz, R. Cochran, and S. Reda, "Thermal monitoring of real processors: Techniques for sensor allocation and full characterization," in *Proc. Design Automation Conf. (DAC)*, 2010.

[37] X. Li, X. Li, W. Jiang, and W. Zhou, "Optimising thermal sensor placement and thermal maps reconstruction for microprocessors using simulated annealing algorithm based on pca," *IET Circuits, Devices Systems*, vol. 10, no. 6, pp. 463–472, 2016.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. http://www.deeplearningbook.org.

[39] J. L. Greathouse and G. H. Loh, "Machine learning for performance and power modeling of heterogeneous systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, IEEE, 2018.

[40] R. G. Kim, J. R. Doppa, and P. P. Pande, "Machine learning for design space exploration and optimization of manycore systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, IEEE, 2018.

[41] K. Zhang, A. Guliani, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman, "Machine learning-based temperature prediction for runtime thermal management across system components," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, pp. 405–419, Feb 2018.

[42] H. Amrouch and J. Henkel, "Lucid infrared thermography of thermally-constrained processors," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 347–352, July 2015.

[43] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[44] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, pp. 90–93, Jan 1974.

[45] Intel, "Intel Performance Counter Monitor (PCM)." https://software.intel.com/en-us/articles/intel-performance-counter-monitor.

[46] AMD, "AMD uProf." https://developer.amd.com/amd-uprof/.

[47] Phoronix, "Open-Source, Automated Benchmarking." https://www.phoronix-test-suite.com/.

[48] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2008.

[49] http://www.spec.org/cpu2000/CFP2000/.

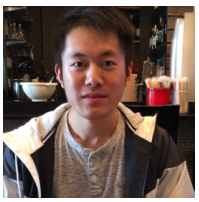
[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[51] Intel, "Technical resources: Intel core processors." https://www.intel.com/content/www/us/en/products/docs/processors/core/core-technical-resources.html.

[52] Y. Zhang, B. Shi, and A. Srivastava, "Statistical framework for designing on-chip thermal sensing infrastructure in nanoscale systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 270–279, 2014.

[53] H. Zhou, X. Li, C. Cher, E. Kursun, H. Qian, and S. Yao, "An information-theoretic framework for optimal temperature sensor allocation and full-chip thermal monitoring," in *DAC Design Automation Conference 2012*, pp. 642–647, 2012.

Sheriff Sadiqbatcha (Student Member, IEEE) received his B.S. degree (most outstanding graduate of the year) in Computer Engineering from California State University, Bakersfield in 2016 and his M.S. degree (magna cum laude) in Electrical Engineering from the University of California, Riverside in 2019. He is currently a Ph.D. student and Associate Instructor in the Department of Electrical and Computer Engineering at the University of California, Riverside. His research interests include pre-silicon reliability (Electromigration), and applied machine-learning in the area of Electronic Design Automation (EDA), post-silicon thermal, power, and reliability modeling and control.



Jinwei Zhang (Student Member, IEEE) is a current PhD student who joined VSCLAB in the Department of Electrical and Computer Engineering at the University of California Riverside in Fall 2018. He received his B.S. in Electrical and Control Engineering at Beijing Institute of Technology, China and M.S. in Electrical Engineering at Washington University in Saint Louis, United States. His research interests are in Electronic Design Automation (EDA), system level thermal and power modeling for VLSI and optimization with modern machine learning techniques.

Hussam Amrouch (Member, IEEE) is a Junior Professor for the Semiconductor Test and Reliability (STAR) chair within the Computer Science, Electrical Engineering Faculty at the University of Stuttgart as well as a Research Group Leader at the Karlsruhe Institute of Technology (KIT), Germany. He received his Ph.D. degree with distinction (Summa cum laude) from KIT in 2015. His main research interests are design for reliability and testing from device physics to systems, machine learning, security, approximate computing, and emerging technologies with a special focus on ferroelectric devices. He holds seven HiPEAC Paper Awards and three best paper nominations at top EDA conferences: DAC'16, DAC'17 and DATE'17 for his work on reliability. He currently serves as Associate Editor at Integration, the VLSI Journal. He has served in the technical program committees of many major EDA conferences such as DAC, ASP-DAC, ICCAD, etc. and as a reviewer in many top journals like T-ED, TCAS-I, TVLSI, TCAD, TC, etc. He has 115 publications (including 43 journals) in multidisciplinary research areas across the entire computing stack, starting from semiconductor physics to circuit design all the way up to computer-aided design and computer architecture. He is a member of the IEEE. ORCID 0000-0002-5649-3102.

Sheldon X.-D. Tan (Senior Member, IEEE) received his 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, in 1999. He is a Professor in the Department of Electrical Engineering, University of California, Riverside, CA. He also is a cooperative faculty member in the Department of Computer Science and Engineering at UCR. 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, parallel computing and adiabatic and Ising computing based on GPU and multicore systems. He has published more than 320 technical papers and has co-authored 6 books on those areas.

Dr. Tan received NSF CAREER Award in 2004. He received Best Paper Awards from ICSICT'18, ASICON'17, ICCD'07, DAC'09. He also received the Honorable Mention Best Paper Award from SMACD'18. He was a Visiting Professor of Kyoto University as a JSPS Fellow from Dec. 2017 to Jan. 2018. He is serving as the TPC Chair for ASPDAC 2021, and the TPC Vice Chair for ASPDAC 2020. He is serving or served as Editor in Chief for Elsevier's Integration, the VLSI Journal, the Associate Editor for three journals: IEEE Transaction on VLSI Systems (TVLSI), ACM Transaction on Design Automation of Electronic Systems (TODAES), Elsevier's Microelectronics Reliability and MDPI Electronics, Microelectronics and Optoelectronics Section.