Machine Learning Based Online Full-Chip Heatmap Estimation

Sheriff Sadiqbatcha*, Yue Zhao*, Jinwei Zhang*, Hussam Amrouch†, Jörg Henkel†, Sheldon X.-D. Tan*,
* Department of Electrical and Computer Engineering, University of California, Riverside, CA, USA
† Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

***Abstract*—Runtime power and thermal control is crucial in any modern processor. However, these control schemes require accurate real-time temperature information, ideally of the entire die area, in order to be effective. On-chip temperature sensors alone cannot provide the full-chip temperature information since the number of sensors that are typically available is very limited due to their high area and power overheads. Furthermore, as we will demonstrate, the peak locations within hot-spots are not stationary and are very workload dependent, making it difficult to rely on fixed temperature sensors alone. Therefore, we propose a novel approach to real-time estimation of full-chip transient heatmaps for commercial processors based on machine learning. The model derived in this work supplements the temperature data sensed from the existing on-chip sensors, allowing for the development of more robust runtime power and thermal control schemes that can take advantage of the additional thermal information that is otherwise not available. The new approach involves 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, we apply Long-Short-Term-Memory (LSTM) neutral networks with system-level variables such as chip frequency, instruction counts, and other performance metrics as inputs. 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. Our study shows that only 6×6 DCT coefficients are required to maintain sufficient accuracy across a variety of workloads. Experimental results show that the proposed approach can estimate the full-chip heatmaps with less than 1.4°C root-mean-square-error and take only ∼19ms for each inference which suits well for real-time use.**

I. 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], and hence has become one of the leading concerns in the industry today. To address this trend, runtime power and thermal control schemes are being implemented in most, if not all new generations of processors [2], [3]. These control schemes depend on accurate real-time temperature information of the entire die area in order to be effective [4], [5].

On-chip temperature sensors alone cannot provide the full-chip temperature information since the number of sensors that can be placed on a chip is limited due to their high area and power overheads. A better solution is to develop thermal models that can estimate the temperature distribution across the entire chip during runtime [6]–[8]. The model can then be used to supplement the data from the few on-chip sensors, expanding the overall thermal information available during runtime. However, accurate online estimation of full-chip heatmaps is challenging, especially when it comes to commercial off-the-shelf multi-core processors.

This work is supported in part by NSF grant under No. CCF-1527324, and in part by NSF grant under No. CCF-1816361 and in part by NSF grant under OISE-1854276.

978-1-7281-4123-7/20/$31.00 ©2020 IEEE

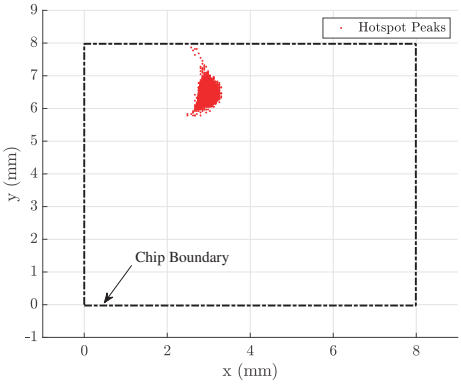


Fig. 1. Peak locations of a single hotspot on an Intel i5-3337U

Full-chip thermal characterization and estimation have been well studied in the past. In general, two kinds of strategies have been explored. One strategy is based on the placement of on-chip temperature sensors and reconstruction of the full-chip heatmaps from the sensor data [9]–[11]. However, these methods need to find the optimum number of sensors and their locations. This still remains a challenging task as sensors incur significant overheads and they cannot be inserted in arbitrary locations. The overheads and thermal estimation accuracy has always been a trade-off. Furthermore, the localized thermal hot-spots are often not stationary. The thermal peak migrates significantly depending on the workload that the processor is under. For example, Fig. 1 shows the peak locations (from experimental data) of a single hot-spot at different time instances. As a result, accurate heatmaps with exact hot-spot locations are very difficult to estimate based on fixed thermal sensors alone (unless if the sensors are uniformly placed as shown in [9], which is difficult if not impossible to achieve given the design restrictions).

The second strategy is to estimate the full-chip heatmaps from a thermal model and power related information. These thermal models can be built using the so-called "bottom-up" approaches such as HotSpot [6] based simplified finite difference methods, finite element methods [12], equivalent thermal RC networks [13], and recently proposed behavioral thermal models based on the matrix pencil method [14] and the subspace identification method [15], [16]. However, the existing thermal modeling methods still have a few drawbacks. First, many need accurate power-traces as inputs; but estimating the power of each functional unit (FU) of a real processor under varying workloads is not a trivial task, if not infeasible [17], [18]. On the other hand, from the system-level thermal or power management perspective, the parameters that can be easily accessed are core frequency, voltage, and many other performance metrics natively supported by most

commercial processors [19]. Thermal models that are functions of these parameters will be more desirable and practical. Second, it is difficult to calibrate these models for practical use due to simplified modeling, boundary conditions, and the lack of sufficient accuracy. Lastly, most such models employ numerical methods to solve, which are not suitable for real-time use. In contrast, neural network based models are much more lightweight, making them ideal for online inference.

A machine learning based method has been proposed in the past to predict the future temperature of the chip based on the current on-chip temperature sensor data [20]. However, this method lacks spatial resolution as the prediction is limited to the locations on the chip where temperature sensors are present. As previously mentioned, the number of temperature sensors available in the chip is extremely limited due to their area and power overheads. Case in point, in [21] the authors identify 18 hot-spots on a commercial processor that only has 2 on-chip temperature sensors. Hence, a new technique capable of real-time estimation of the temperatures across the entire chip area is more desirable. A prior attempt at this goal showed promising results [22], [23], however this method has couple of drawbacks. First, the model only works on a predetermined set of applications, which is not realistic for general purpose processors. Second, the model is validated using only the on-chip temperature sensor data, not measured full-chip heatmaps. Third, the model had an inference time in the order of seconds, which is too slow for use with thermal/power control algorithms which are typically reactive in nature. Ideally, we want inference times in the order of milliseconds to match the thermal time constant of semiconductor chips.

To overcome the aforementioned challenges, we propose a novel approach for real-time estimation of full-chip heatmaps for commercial microprocessors based on machine learning. This work exploits the correlation between the processor's temperature and system-level variables such as frequency, voltage, and other performance metrics that have been shown to be effective in the past [24]–[26]. However, the existing methods relied on manually identifying the low-level performance metrics that are correlated with each functional-unit (FU) on the chip. Additionally, for a FU-wise prediction, at least one low-level performance metric must be recorded for each FU at all times. However, only a hand full of performance counting registers are available in the processor (typically about 10), hence a full-chip estimation is not feasible using this method. Alternatively, in this work we use high-level performance metrics that give us a comprehensive view of the processor's utilization in real-time. The correlation between the transient behavior of high-level performance metrics and the processor's thermal profile is automatically learned. This work is inspired by the recent rapid advancements in Recurrent Neural Networks (RNN) for time-series system modeling. In our method, we first obtain the accurate spatial and temporal heatmaps of a commercial microprocessor using a lucid infrared thermal imaging setup, while nominal working conditions are maintained on the chip. We then compress the heatmaps using 2D Discrete Cosine Transformation (DCT) such that they can be expressed with just their dominant DCT frequencies. This greatly reduces the dimensionality of the model, consequently improving accuracy and performance. To build the dynamic thermal model, we apply Long-Short-Term-Memory (LSTM) networks with system-level variables such as chip frequency, instruction counts and other performance metrics as inputs. Once trained, the model will be capable of real-time estimation of full-chip heatmaps. Experimental results validated with measured thermal data show that the proposed approach can estimate the full-chip heatmaps with less than 1.4°C root-mean-square (RMS) error with approximately 19ms of processing time for each inference.

II. Proposed Approach

A brief overview of the proposed approach will be presented in this section, along with a description of the thermal setup used for collecting the necessary data from a commercial microprocessor while it is under load.

A. Overview

The proposed approach requires two critical pieces of data that has to be collected in synchronous while the processor is under load. First is the time-stamped heatmaps collected from the chip at a steady sampling rate, and second is a suite of high-level performance metrics captured at the same time instances as the temperature data. To this end, we have built an advanced IR thermography setup that records lucid heatmaps of the processor under test. This setup will be discussed in detail in the next subsection. At the same time, a suite of high-level performance metrics will be recorded in synchronous with the capture rate of the IR camera. Once sufficient data is acquired, a specialized Recurrent-Neural-Network (RNN) architecture called Long-Short-Term-Memory (LSTM) network will be employed to train the online thermal model. Fig. 2(a) illustrates the data preparation and training phase of the proposed approach. Once trained, the thermal model will use the performance metrics as inputs to estimate the full-chip heatmaps in real-time. The testing phase of the proposed approach is illustrated in Fig. 2(b). Each step illustrated in Fig. 2 will be discussed in detail in the following sections.

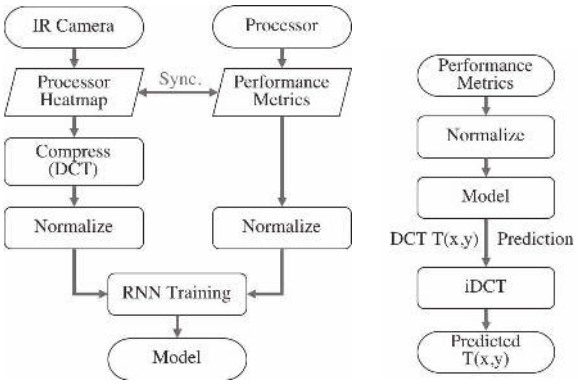
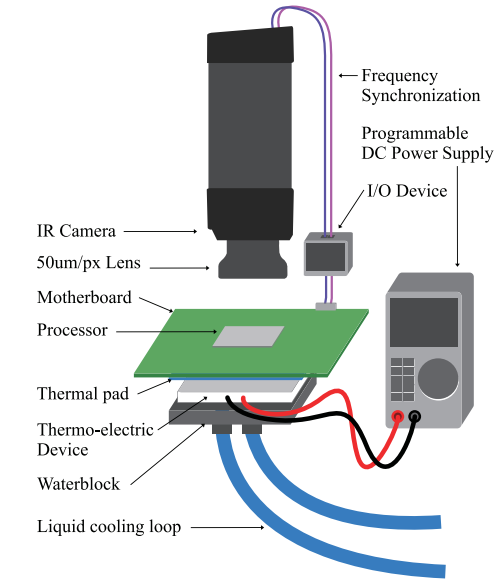


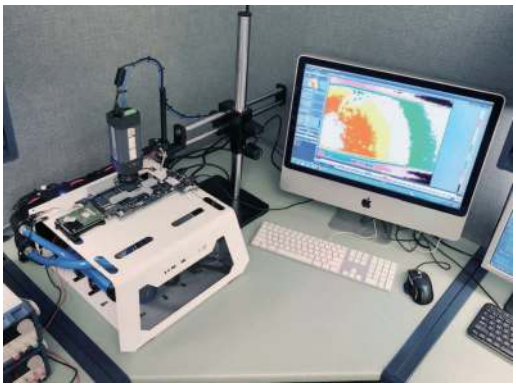
Fig. 2. Proposed algorithm flow: (a) Training (b) Testing.

B. Our IR Thermography Setup

As mentioned, the proposed approach relies on accurate time-series data of the full-chip heatmaps measured directly from the processor under test. Externally measuring this information is challenging, especially when the processor is under load as it requires the processor to be operated without the traditional front-mounted cooling systems (i.e. heat-sink). To address this issue, we have built an advanced infrared (IR) thermography setup shown in Fig. 3. This setup is based on the thermal imaging setup proposed in [27]. The setup features a thermo-electric device mounted on the PCB directly beneath the processor allowing it to be cooled from underneath; leaving the front side fully exposed to the IR camera without any interference layer in-between. Unlike the traditional oil-based front-cooling methods, no de-embedding [18] is required in our setup. Instead, a programmable 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 using a heat-sink [27].



(a)



(b)

Fig. 3. Our IR thermography setup [21]

Detailed description of the IR thermography setup is as follows. The thermal camera used in this setup is a FLIR A325sc (16-bit 320×240px, 60Hz). The camera is rated for the temperature range of $0°C$ to $328°C$, and spectral range of $7.5\mu m$ to $13\mu m$. In order to enhance the spatial resolution of the camera, a microscope lens is used providing a spatial resolution of $50\mu m$/px. The IR camera has an internal waveform generator that outputs a square waveform in synchronous with the capture frequency of the camera. An I/O device is then employed to interface the waveform generator to the processor under test, so that the performance metrics (recorded on the processor) can be synchronized with the thermal data recorded by the IR camera. The processor under test is an Intel i5-3337U, which has 2 cores with 2 threads per core. Mounted on the PCB directly underneath the processor is the thermo-electric-based cooling system which includes a Peltier device powered by a programmable power source. A liquid cooling loop is used to cool the hot-side of the Peltier device and ensure proper operation.

III. Data Preparation

As previously mentioned, the two critical pieces of data that we use in this work are the heatmaps and performance metrics recorded in synchronous with each other. The end goal is to train a machine learning model, which uses the performance metrics as inputs to estimate the heatmaps in real-time. As with any machine learning approach, proper preparation and normalization of the training datasets is of utmost importance. The compression and normalization of the datasets prior to training the LSTM network will be discussed in this section.

A. Performance Metrics

Since an Intel processor is used in this study, the performance metrics data is collected using Intel's Performance Counting Monitor (PCM) [28], which is a tool that provides high-level runtime performance metrics for Intel processors. Table I shows the complete list of 80 performance metrics that we collect from the Intel i5-3337U. As we will discuss in detail in Section IV, our RNN architecture uses the hyperbolic tangent ($Tanh$) activation function in the LSTM layers, and the linear activation in the final dense layer. In order to take full advantage of the effective range of the $Tanh$ activation function, all 80 input PCM metrics will be normalized to the range of $[-1, 1]$ using the min-max normalization scheme given in Eq.(1).

$$PCM'_i = \left(\frac{PCM_i - min(PCM)}{max(PCM) - min(PCM)} \times 2\right) - 1 \quad (1)$$

TABLE I
Performance Metrics (Intel PCM)

Pkg.	Pkg.	Core1.1	Core1.2	Core2.1	Core2.2
exec	inst nom	exec	exec	exec	exec
IPC	inst nom%	IPC	IPC	IPC	IPC
freq	C2res%	freq	freq	freq	freq
afreq	C3res%	afreq	afreq	afreq	afreq
L3 miss	C6res%	L3 miss	L3 miss	L3 miss	L3 miss
L2 miss	C7res%	L2 miss	L2 miss	L2 miss	L2 miss
L3 hit	energy (J)	L3 hit	L3 hit	L3 hit	L3 hit
L2 hit	temp	L2 hit	L2 hit	L2 hit	L2 hit
L3 MPI		L3 MPI	L3 MPI	L3 MPI	L3 MPI
L2 MPI		L2 MPI	L2 MPI	L2 MPI	L2 MPI
read rate		C0res%	C0res%	C0res%	C0res%
write rate		C1res%	C1res%	C1res%	C1res%
inst count		C3res%		C3res%	
ACYC		C6res%		C6res%	
physIPC		C7res%		C7res%	
physIPC%		temp		temp	

B. Heatmap Compression

Heatmaps of the Intel i5-3337U captured using our IR thermography setup have a image resolution of 177x166 pixels. This constitutes to a total pixel count of 29382. It is not practical for a machine learning model to directly estimate the entire heatmap as this would require the output dimensionality of 29382 in order to achieve a pixel-wise estimation. As a result, we need to compress the heatmaps such that they can be expressed using just their dominant features instead.

To this end, we use 2D Discrete Cosine Transformation (DCT) to convert the heatmaps, $T(x, y)$, into spatial frequency domain [29]. This allows us to extract the dominant low-frequency components of the heatmaps and train our machine learning model to estimate only these dominant frequencies. Inverse DCT can 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" [29] as 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

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

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 1-D DCT on each column and then on each row of the matrix.

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 of our processor 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 $\mathbf{6 \times 6}$ DCT coefficients as increasing the number of features further produces marginal benefits. *With this compression, the output of the model only needs the dimensionality of 36, instead of 29382.*

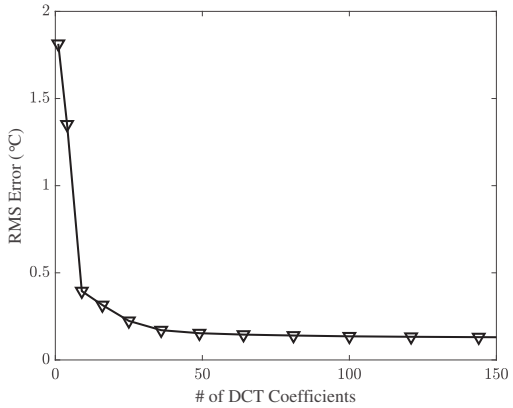


Fig. 4. RMS error between actual heatmaps and compressed heatmaps using varying number of DCT coefficients

Similar to the performance metrics, the DCT coefficients from the heatmaps are also normalized to the range of $[-1, 1]$ prior to training. Once the model has been trained, the output of the model will be denormalized before the inverse DCT operation (iDCT) is performed to construct the estimated heatmap. Similar to its forward counterpart, the 2D iDCT consists of two separate 1D iDCT steps (3) on the rows and columns respectively.

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

IV. LSTM-Based Estimation Model

From the machine learning perspective, our task in this work is a time-series prediction problem. Hence, a variant of recurrent-neural-networks (RNN) known as long-short-term-memory-network (LSTM) can be used, as this is a common choice for such applications [30]. The key structure of the LSTM network that we used is diagrammed in Fig. 5. The network mainly consists of five LSTM recurrent layers, with 160, 130, 100, 70, and 48 nodes per layer respectively, and one linear layer, with 36 nodes, used as the output layer. All LSTM layers use the $Tanh$ activation function where as the dense layer uses a linear activation function. Dropout layers are added between adjacent LSTM layers with a 20% dropout rate. This prevents over-fitting during the training phase and hence increases the model's accuracy. The network has an input dimensionality of 80 for the 80 performance metrics, and an output dimensionality of 36 as the network will be trained to estimate the first 36 (6×6) DCT coefficients.

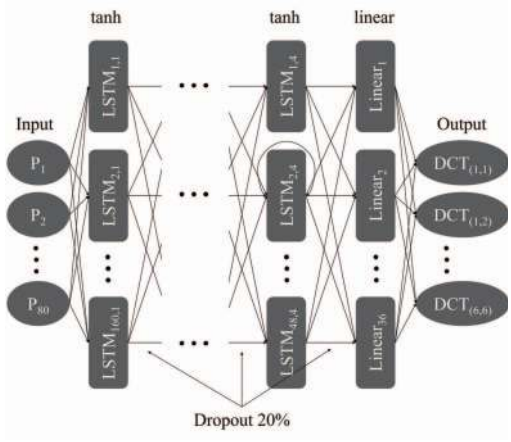


Fig. 5. LSTM network architecture: 5 LSTM Layers (160, 130, 100, 70, and 48 nodes respectively), and 1 Dense Layer (36 nodes)

V. Experimental Results

In our experimental setting, approximately 1 hour of runtime data were collected at the capture frequency of 60Hz. The total data that this method requires will depend on the type of processor and the number, and variety of workloads that is required to thoroughly task all the subsystems of the processor. As a rule of thumb, more data will generally lead to a better performing and more reliable model.

As previously mentioned, each time-step of data consists of 80 PCM metrics and one heatmap of the processor, both captured concurrently using the synchronization method discussed in Section II. During data collection, the processor was subjected to a variety of workloads from the Phoronix benchmark suite [31]. These workloads are generally split into three categories: processor, memory, and system. The processor benchmarks are generally compute intensive, whereas the memory benchmarks are generally memory intensive. The system benchmarks are designed to task all subsystems of the chip. Several benchmarks from each category were selected to ensure the chip was subjected to a wide variety of workloads with varying execution patterns. The specific benchmarks used in this study are listed in Table II. Note, the workloads that we have selected are only meant to demonstrate the proposed approach and show the performance of the model. The model itself is independent of the workloads as it is based on the processor's utilization rate characterized by high-level performance metrics.

TABLE II
Benchmarks

Processor	Memory	System
7zip	Cyclictest	RAMspeed
AObench	Gimp	Stream
FLAC	Git	Ttest
OpenCV	PHPbench	

After the data has been acquired, the method discussed in Section III was used to compress and normalize the data in preparation for training. This process results in 80 normalized PCM metrics, and the first 36 most dominant heatmap DCT frequencies for each time-step. The data used for training and

testing are split as follows. For each 1000 timesteps of data, 640 timesteps were used for training, 160 timesteps were used for validation, and 200 timesteps were put aside for testing. Hence, in total, 64% of the entire dataset was used for training, 16% was used for validation, and 20% was used for testing. The training and testing datasets were kept completely isolated from each other to ensure that none of the testing data was used in the training process, and vice-versa. After several rounds of experimentation and tuning, the network shown in Fig. 5 was selected. The model was implemented in Python using $Keras$ [32], with $Tensorflow$ [33] as its back-end. The network was then trained using a Nvidia Tesla K40c GPU for a total of 100 epochs with 60 time steps per epoch in the LSTM layers.

Once trained, this method results in what effectively is a "black-box" model which can be executed on the chip or integrated into the operating system. In real-time, the model will use the 80 normalized performance metrics as inputs to estimate the first 36 most dominant DCT frequencies. Inverse DCT operation, shown in Eq.(3), can then be used to convert the estimated DCT frequencies into the estimated heatmap for each time-step.

The testing results showed that the model performs exceptionally well. The estimated 36 (6×6) DCT coefficients follow the trends of the measured data with marginal error. The estimated 6×6 DCT coefficients are shown in Fig. 6 plotted along with their measured counterparts from the testing dataset. Fig. 7 shows the measured heatmap alongside the estimated heatmap from a random time-step during the execution of the 7zip benchmark. The error map (actual heatmap - estimated heatmap) is also shown, where the peak error at this time instance is 0.53°C.

To formally compute the overall accuracy of the model, we use the root-mean-square (RMS) equation given in Eq.(4).

$$RMS = \sqrt{\frac{\sum_{t=1}^{tmax} \sum_{x=1}^{n} \sum_{y=1}^{m} (T(x,y,t) - T'(x,y,t))^2}{n \times m \times tmax}} \tag{4}$$

Where T and T' are the measured and estimated heatmaps respectively, t is testing timestep, tmax is the final testing timestep, x and y are spatial coordinates on the heatmap, n = 166, and m = 177 are vertical and horizontal pixel counts respectively. Using Eq.(4) the RMS error calculated for the entire testing dataset is 1.4°C, which is very reasonable for most applications that need real-time information of the temperatures across the entire chip. RMS error computed separately for each benchmark is shown in Table III. As the results show, RMS error for most benchmarks is less than 2°C and a few of them are less than 1°C with an overall RMS error of 1.4°C. In terms of performance overhead, the execution time for the model is approximately 19ms per inference. This makes the proposed full-chip heatmap estimation technique not only practical, but also highly desirable for applications such as online thermal management and task scheduling for commercial multi-core processors.

TABLE III
RMS FOR EACH BENCHMARK

Benchmark	RMS	Benchmark	RMS
7zip	1.19°C	Cyclictest	1.27°C
AObench	0.89°C	Gimp	1.78°C
FLAC	1.59°C	Git	1.12°C
OpenCV	1.11°C	PHPbench	1.23°C
RAMspeed	0.87°C	Stream	0.55°C
Ttest	2.67°C		

VI. Conclusion

In this article, we have proposed a machine learning based approach to real-time estimation of full-chip heatmaps for commercial microprocessors. In this new approach, we obtain accurate spatial and temporal heatmaps using an advanced infrared thermal imaging system. To build the thermal model, we applied Long-Short-Term-Memory (LSTM) networks with system-level variables such as chip frequency, voltage, and instruction count 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 their dominant DCT frequencies. Our study showed that only 6×6 DCT coefficients are required to maintain sufficient accuracy across a variety of workloads. Experimental results show that the proposed approach can estimate the transient heatmaps with less than 1.4°C RMS error and take only ∼19ms for each inference.

References

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

[2] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. *Micro, IEEE*, 32(3):122–134, May 2012.

[3] Michael Taylor. A landscape of the new dark silicon design regime. 33(5):8–19, October 2013.

[4] K. Skadron, M.R. Stan, and et. al. Temperature-aware microarchitecture. In *Proc. Intl. Symp. on Computer Architecture*, 2006.

[5] Joonho Kong, Sung Woo Chung, and Kevin Skadron. Recent thermal management techniques for microprocessors. *ACM Comput. Surv.*, 44(3):13:1–13:42, jun 2012.

[6] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R. Stan. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May 2006.

[7] 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*, 16(1):86–99, 2007.

[8] 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.

[9] Ryan Cochran and Sherief Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 478–483, New York, NY, USA, 2009. ACM.

[10] Juri Ranieri, Alessandro Vincenzi, Amina Chebira, David Atienza, and Martin 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, pages 636–641, New York, NY, USA, 2012. ACM.

[11] 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*, 10(6):463–472, 2016.

[12] Siva P. Gurrum, Yogendra K. Joshi, William P. King, Koneru Ramakrishna, and Martin Gall. A compact approach to on-chip interconnect heat conduction modeling using the finite element method. *Journal of Electronic Packaging*, 130:031001.1–031001.8, September 2008.

[13] Y. C. Gerstenmaier and G. Wachutka. Rigorous model and network for transient thermal problems. *Mircroelectronics Journal*, 33:719–725, September 2002.

[14] Duo Li, Sheldon X.-D. Tan, Eduardo H. Pacheco, and Murli Tirumala. Parameterized architecture-level dynamic thermal models for multicore microprocessors. *ACM Trans. Des. Autom. Electron. Syst.*, 15(2):1–22, 2010.

[15] 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.

[16] 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*, 47(1), January 2014. in press, online access: http://www.sciencedirect.com/science/article/pii/S0167926013000412.

[17] 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*, 12(3):1–29, 2007.

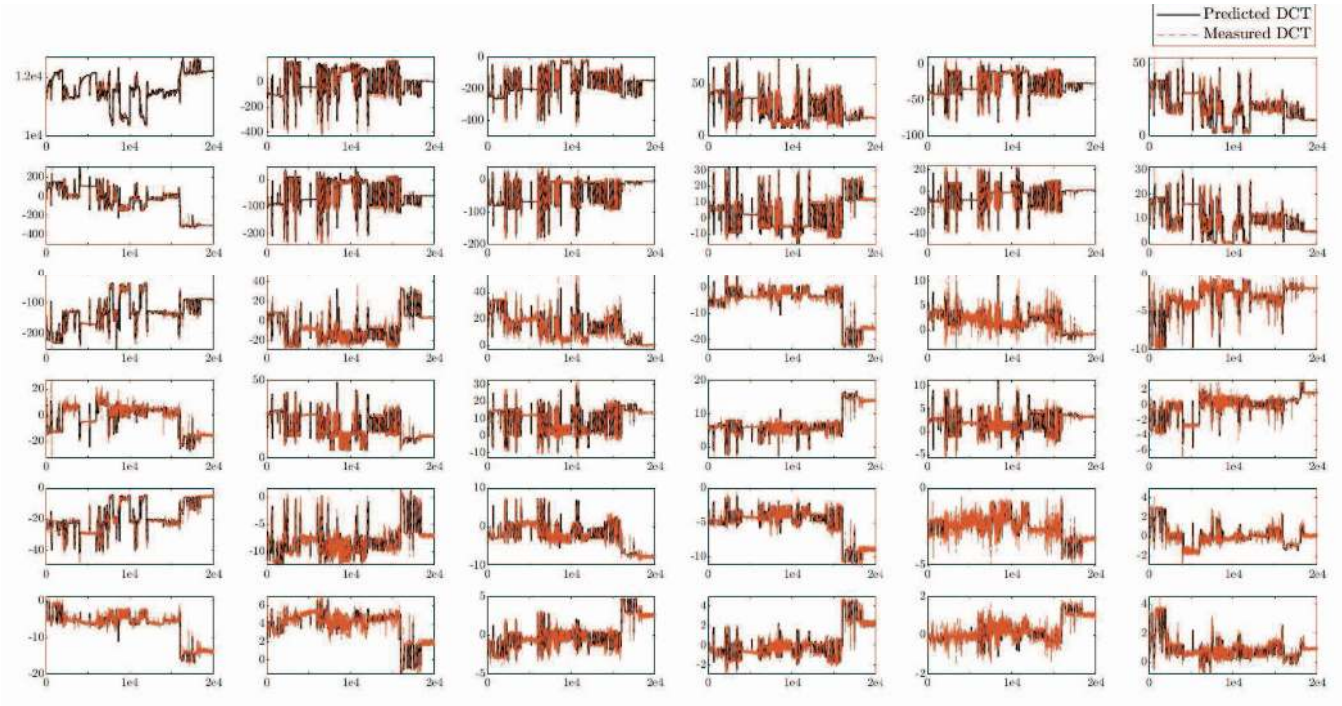
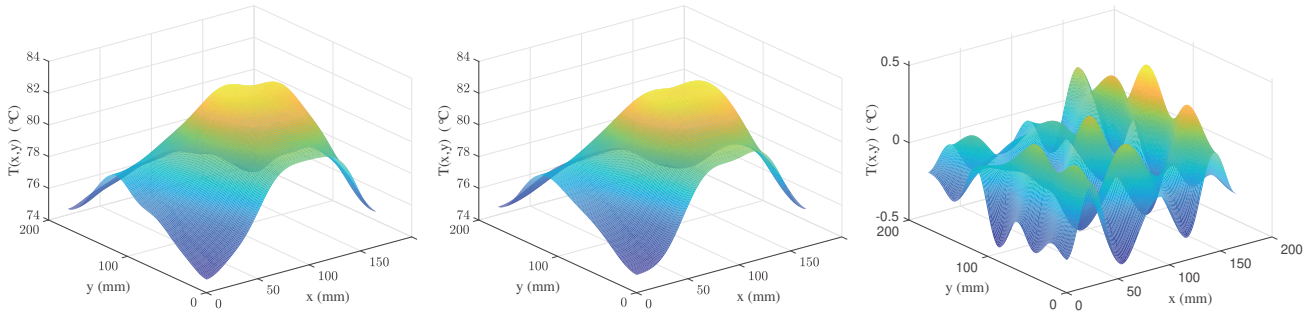


Fig. 6. Estimated vs measured 6x6 DCT coefficients (x-axis: time, y-axis: DCT amplitude)



(a) (b) (c)

Fig. 7. (a) Measured heatmap from the 7zip benchmark. (b) Estimated heatmap for the 7zip benchmark (c) Error map (measured - estimated heatmap).

[18] 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)*, pages 39–44, Sept 2013.

[19] 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*, 29(2):405–419, Feb 2018.

[20] Federico Pittino, Roberto Diversi, Luca Benini, and Andrea Bartolini. Robust online identification of thermal models for in-production HPC clusters with machine learning-based data selection. *CoRR*, abs/1810.01865, 2018.

[21] 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.

[22] Devendra Rai, Hoeseok Yang, Iuliana Bacivarov, and Lothar Thiele. Power agnostic technique for efficient temperature estimation of multicore embedded systems. In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, CASES '12, pages 61–70, New York, NY, USA, 2012. ACM.

[23] D. Rai and L. Thiele. A calibration based thermal modeling technique for complex multicore systems. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1138–1143, March 2015.

[24] 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*, pages 8 pp.–, April 2005.

[25] J. S. Lee, K. Skadron, and S. W. Chung. Predictive temperature-aware dvfs. *IEEE Transactions on Computers*, 59(1):127–133, Jan 2010.

[26] 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)*, pages 1215–1220, March 2013.

[27] 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)*, pages 347–352, July 2015.

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

[29] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, Jan 1974.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

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

[32] François Chollet et al. Keras, 2015. https://keras.io.

[33] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.