**Proceedings of the ASME 2022 on
Packaging and Integration of Electronic and Photonic Microsystems
INTERPACK2022
October 25-27, 2022, Garden Grove, CA**

INTERPACK2022-96751

MACHINE LEARNING AND SIMULATION BASED TEMPERATURE PREDICTION ON HIGH-PERFORMANCE PROCESSORS

Carlton Knox[1],*, Zihao Yuan[1], and Ayse K.Coskun[1]

[1]Boston University, Boston, MA

ABSTRACT

Emerging thermal management policies for high-power processors often rely on the temperature readings from on-chip digital thermal sensors. However, thermal sensors may not accurately measure the maximum temperature on chip. This is because thermal hot spots are typically located near important CPU components, limiting the power and physical space available for thermal sensors. As a result, sensors usually need to be placed some distance away from the hot spots. Additionally, on-chip thermal sensors also operate within an error margin, which could under/over-estimate the temperature readings. Prior methods introduced machine learning algorithms for predicting chip temperatures trained with Infrared (IR) camera measurements of the physical chip to construct accurate on-chip thermal profiles. While such methods produce an accurate model, the thermal imaging setup is expensive, and it can be time-consuming to collect and process the temperature data for a physical chip. This paper proposes a simulation-based method of using a machine learning regression model to predict a chip's full temperature map based solely on the current power usage, core utilization, and measured sensor temperatures. The proposed model is trained and evaluated based on data generated from performance, power, and thermal simulations for the Intel i7 6950× Extreme Edition processor. When running a set of realistic benchmarks, this model is able to accurately predict temperatures within a root mean squared error (RMSE) of less than 0.25°C. The proposed model's accuracy is not affected by the placement of the thermal sensors, and the maximum error resulting from the placement of thermal sensors is less than 0.12°C. For a real-world application, the proposed model can be trained based on realistic simulation or measured temperature data, then be applied to predict a chip's temperature map in real-time. Using actual temperature data measured from an IR camera is more accurate, but the IR camera setup itself is expensive. Using simulation data to train the machine learning model is low-cost and more practical than temperature prediction based on an expensive IR camera.

*Corresponding author

Keywords: Temperature prediction, Machine learning, Linear regression, High-performance processor

NOMENCLATURE

IR	Infrared
$PACT$	A standard cell level to architectural level parallel compact thermal simulator
$Sniper$	A parallel, high-speed and accurate x86 simulator
$McPAT$	An integrated power, area, and timing modeling framework for multicore and manycore architectures
NAS	NASA Advanced Supercomputing
bt	Block Tri-diagonal solver
cg	Conjugate Gradient
dc	Data Cube
ep	Embarrassingly Parallel
ft	Discrete 3D Fast Fourier Transform
is	Integer Sort, random memory access
lu	Lower-Upper Gauss-Seidel solver
mg	Multi-Grid on a sequence of meshes
sp	Scalar Penta-diagonal solver
ua	Unstructured Adaptive Mesh
TDP	Thermal design power (W)
CV	Cross-validation
$LOOCV$	Leave one out cross-validation
$R2$	Coefficient of determination

1. INTRODUCTION

High chip temperatures have been a primary concern for several decades. Localized hot spots resulting from these high power densities not only decrease the lifetime of processors [1] but also increase transistor delays as well as leakage power [2]. In addition, the heterogeneity in on-chip heat distribution incurred by these hot spots is expected to become more severe with the integration of heterogeneous architectures on a single die, such as a collection of CPUs, GPUs, accelerators, and FPGAs. To enhance reliability, researchers have proposed runtime policies that use control knobs such as dynamic voltage and frequency scaling, task scheduling, and thread migration (e.g., [3]). Modern processors utilize digital thermal sensors to track the processor's temperature at various strategic locations to manage runtime temperatures. However, on-chip thermal sensors may not accurately measure the temperature profile and maximum temperature on-chip. We identify three major challenges in accurately obtaining the temperature profile and hot spot temperatures using thermal sensors. First, because of the placing and routing difficulties, thermal sensors may not be placed at the exact location of the hot spots, leading to under-estimating the hot spot temperatures and potentially changing the dynamic thermal runtime policy decision [4]. Second, the spatial and temporal fluctuations in thermal hot spots due to workload behavior make tracking the hot spot temperature on-chip particularly challenging [4, 5]. Third, on-chip thermal sensors operate within an error margin, which could under/over-estimate the temperature readings by $\pm 1°C$ [6, 7].

To reconstruct accurate on-chip thermal profiles, a recent body of work has introduced using machine learning models to predict chip temperatures trained with infrared (IR) camera measurements of the physical chip [5, 8]. Other works investigate how to intelligently place the thermal sensors on-chip to perform accurate thermal profile monitoring via IR camera measurements [4, 9]. While existing methods produce accurate temperature results, the expensive IR camera setup makes these methods hard to implement broadly by the research community. In addition, additional steps of collecting and processing data from the IR camera measurements make this method complex and time-consuming. Previous work has also introduced a simulation-based method to mitigate the inaccuracies of the on-chip thermal sensors based on analytical models [6]. However, this method targets estimating the temperature at locations of interest instead of regenerating the full thermal profile. Predicting the full thermal map is essential since being able to identify both the hot and cold spots on-chip can benefit the runtime policies such as task allocation and scheduling to achieve better chip performance under temperature constraint [10]. In addition, the locations of interest are challenging to determine, given the different behaviors of the workload.

This paper proposes a simulation-based method of using a machine learning regression model to predict a chip's full temperature profile based solely on the current total power usage of the chip, workload-core mappings, and measured thermal sensors temperatures. We train and validate the proposed machine learning model based on data generated from architectural performance, power, and thermal simulations of an Intel i7 6950× processor. We observe that, using the proposed method with simulation data, it is possible to train a highly accurate machine learning regression model. In addition, the proposed simulation-based method can generally be applied to many processor designs without necessitating an expensive thermal camera setup. The main contributions of the paper are as follows:

- We introduce a machine learning and simulation-based temperature prediction method that accurately predicts the temperature profiles of given chips. We evaluate the accuracy and practicality of the proposed method using an Intel i7 6950× processor running with realistic benchmark applications. Intel i7 6950× is a ten-core desktop processor with a thermal design power (TDP) of 140 W.
- We evaluate the impact of the location of the thermal sensors on our proposed method's accuracy. Experimental results confirm that the thermal sensor placements have minimal effect on the accuracy of the machine learning model, with an RMSE of less than $0.12°C$.
- Our results, including 5-fold CV, show that the machine learning model trained with simulation data achieves high accuracy with an RMSE of less than $0.07°C$. The leave one out cross-validation (LOOCV) results show that our proposed method is able to predict accurate thermal maps for unseen applications and the number of enabled cores, with an accuracy loss of less than $0.25°C$.

2. MODELING AND METHODOLOGY

In this section, we first provide an overview of our methodology for generating a realistic training dataset for the machine learning model through architectural performance, power, and thermal simulators. Then, we discuss the proposed linear regression machine learning model as well as the training and validation methodologies. We use the Intel i7 6950× processor [11] as our target processor and we run ten different applications from the NAS parallel benchmarks [12]. The floorplan of the Intel i7 6950× is shown in Figure 1.

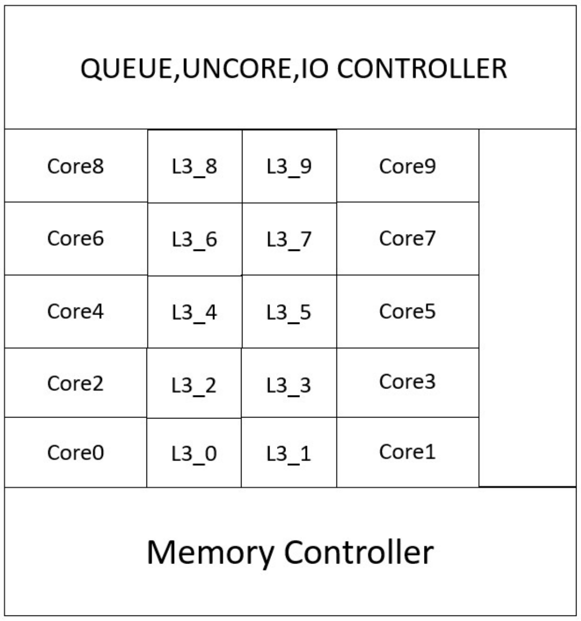

FIGURE 1: INTEL I7 6950× FLOORPLAN.

2.1 Training Data Preparation

In order to generate realistic temperature data for the Intel i7 6950×, we first use the architectural power and performance simulators, Sniper [13], and McPAT [14], to simulate power usage for a set of realistic benchmark applications. We use Sniper because it uses interval-based simulation to model long-running benchmarks while being much faster than cycle-accurate simulation [13]. McPAT simulates the proportional power utilization of

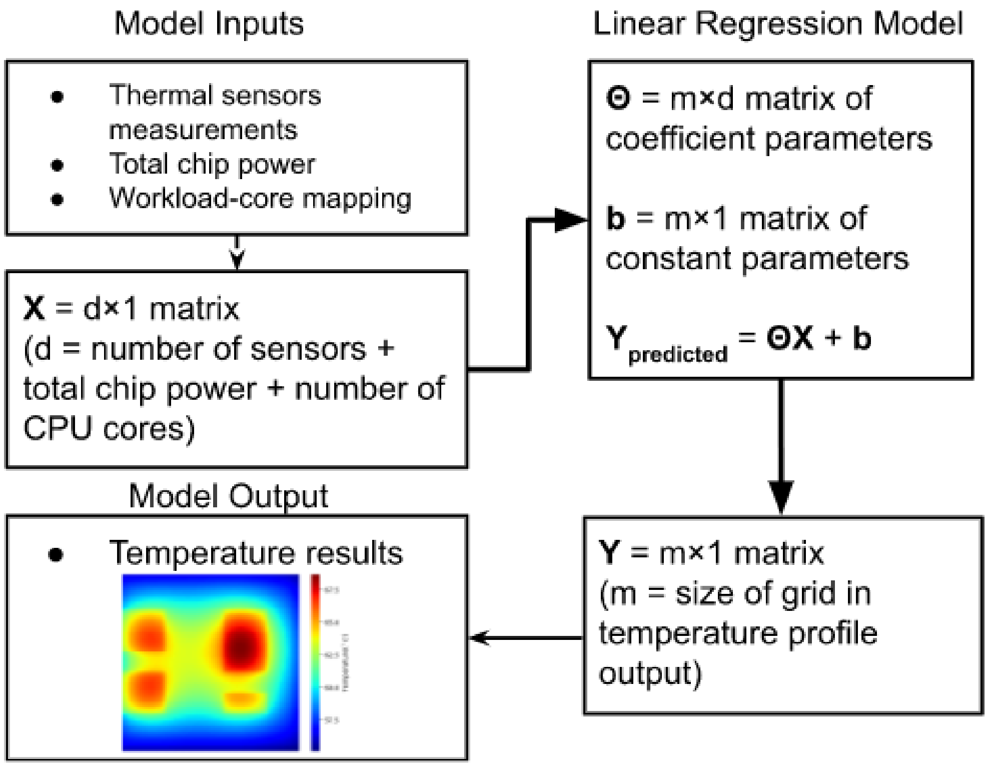
Model Inputs
Thermal sensors measurements
Total chip power
Workload-core mapping
X = d×1 matrix
(d = number of sensors + total chip power + number of CPU cores)
Linear Regression Model
Θ = m×d matrix of coefficient parameters
b = m×1 matrix of constant parameters
Y predicted = ΘX + b
Y = m×1 matrix
(m = size of grid in temperature profile output)
Model Output
Temperature results

FIGURE 2: DIAGRAM OF THE MACHINE LEARNING MODEL.

the CPU core and uncore components (e.g., prior work reported less than 25% error [14]). To ensure our machine learning model is accurate for any workload or configuration of the CPU, we must ensure that the model has seen a wide range of workloads and applications. To generate this realistic set of training data, we select ten applications from the NAS parallel benchmarks: bt, cg, dc, ep, ft, is, lu, mg, sp, and ua. We map the application to a different number of cores (1-10) with different workload-core mapping policies for each application. For example, we can map application cg to 5 cores with a workload-core mapping policy of cores 1, 3, 4, 8, and 9. There are 1023 possible workload-core mappings for a ten-core CPU for each application. Note that we only consider running one application for Intel i7 6950× at a time. If a core does not receive any workload, we set the core to an idle state. We select a random subset containing 36 workload-core mappings to generate the training data.

We run all ten applications from the NAS parallel benchmarks in Sniper for each workload-core mapping and then run McPAT to simulate the power usage. Finally, we extract the power traces, which contain the power values for CPU components such as cores, cache, and IO controllers, from the McPAT. We use PACT [15, 16] as the thermal simulator. PACT is a SPICE-based compact thermal simulator that demonstrates a maximum error of 2.77% for steady-state simulation when compared to finite-element method-based simulators such as COMSOL. The power traces generated using Sniper and MCPAT can be used directly as inputs to PACT. The original power traces generated from Sniper and McPAT are transient. To simplify the inputs of the machine learning model, we average the power traces over the time steps for each application and workload-core mapping to generate steady-state power profiles. The total number of the generated power profiles is 360. The power profiles are then scaled based on the TDP of Intel i7 6950×, which is 140 W, to keep the model within a realistic range. To scale the power profile, we first take the maximum total power of all the power profiles. We then calculate the scaling factor by dividing the TDP by the maximum total power of all the power profiles and then multiply all the function blocks power values by this scaling factor. The resulting scaled power profiles are used as the inputs to PACT and the machine learning model.

In order to simulate the thermal behavior of the chip in PACT, we need to model the floorplan and power profile of the chip. The floorplan of the chip describes the dimensions, locations, and thermal material properties of the CPU's physical functional blocks (e.g., CPU cores or cache blocks). The input power profile for each simulation run describes the power utilization of each functional block. PACT first divides the power profile into a power grid matrix using a predefined grid resolution. It then uses this information to simulate the amount of heat generated by each grid and block and the heat flow between the CPU and cooling layers. For this dataset, we create the power traces with Sniper and McPAT, average and scale the power traces into steady-state power profiles, and then run PACT simulation in steady-state grid mode with a grid resolution of 64×64 and an ambient temperature of $45°C$. We select 64×64 as the grid resolution so that the temperature data points have a significantly finer granularity than the architectural block size. The output of the PACT simulation for each run is a 64×64 temperature grid matrix. This way, each power profile corresponds to a single temperature grid matrix. For the cooling method, we use PACT's medium-cost heat sink [15], which has a size of 40×40 mm^2, and a heat spreader with a size of 20×20 mm^2. The chip layer has a thickness of 0.1 mm and a physical dimension of 14.6×16.8 mm^2. For our set of 360 power profiles, we run the same chip stack using each power profile in PACT to generate 360 corresponding temperature grid matrices to be used in our machine learning model.

2.1.1 Machine Learning Model. The goal of our machine learning model is to predict the full temperature grid matrix (temperature profile) of a CPU based on the power and temperature metrics available at system runtime. The model's input values are the total power usage of the chip at the time of prediction, the temperature values reported by on-chip thermal sensors, and the CPU workload-core mapping, a binary value based on one-hot encoding for each core that represents whether that core is in use. For example, if cores 1, 2, 3 and 7 are in use, the corresponding values would be 0, 1, 1, 1, 0, 0, 0, 1, 0, 0. The model's output data is an array of temperature values corresponding to the 64×64 temperature grids. For the temperature values reported by on-chip thermal sensors, we randomly select ten temperatures from the temperature grid matrix obtained by performing thermal simulation on each power profile and report these temperatures as the readings from thermal sensors. The flow diagram of the temperature profile prediction is shown in Figure 2.

In Figure 2, the inputs and outputs of the machine learning model can be represented as matrices X and Y, where X is a combination of on-chip thermal sensors measurements, total chip power, and binary CPU workload-core mapping, and Y represents the output temperature grid matrix (e.g., 64×64 temperature grid matrix). Since we predict temperature values based on thermal sensors readings and the chip's total power, we select a linear regression model, where each independent predictor models a temperature grid node. Each predictor comprises a set of coefficients corresponding to each input value and a constant. In the linear regression model, the output $Y_{predicted}$ can be directly calculated using $\Theta X + b$, where Θ is the coefficient parameters matrix, and b is the matrix of constant parameters. The input on-chip thermal sensors measurements can be extracted from the PACT output temperature grid matrix by using the temperatures at the grid locations of the thermal sensors. We evaluate the accuracy of the linear regression model for a range of ten randomly selected thermal sensor locations used as input in the next section.

We use the Scikit Learn library for Python and the LinearRegression class for our model training and evaluation. We train the model by splitting the simulation data into training and testing data and then evaluating the trained model's accuracy with the testing data. To validate the model's accuracy, we first perform a 5-fold CV, where the data is split evenly into five randomized buckets. For each bucket, we train the model using the other 80% of the data, test using that 20% set of data, and record the accuracy for each fold. To split and evaluate the data, we use Scikit Learn's cross_val_score function with cv set to 5. Lastly, we perform LOOCV on our model to evaluate its accuracy by excluding applications and core allocations from the training data. We divide the dataset into buckets based on the applications or the number of enabled cores to run the application, then perform n-fold CV. This LOOCV aims to test if the model is still accurate for the applications and workload-core mappings that were not included in the training set, ensuring that the machine learning model can be trained without extensive applications and datasets, and be applied generally. The results of these cross-validations are in the next section.

3. RESULTS AND DISCUSSION

In this section, we validate the power scaling of the simulation data used for training our model and the model itself. We evaluate the model's performance based on different cross-validation methods and discuss how qualitative properties of the dataset, such as average power, affect the model's accuracy.

3.0.1 Training Data Evaluation. As detailed in the previous section, the training data for the machine learning model is generated through the combined use of the architectural performance and power simulators, Sniper and McPAT, and the thermal simulator PACT. With Sniper and McPAT, we are able to simulate the relative power utilization of the different CPU functional blocks. However, since McPAT lacks awareness of some of the implementation details of the CPU architecture, the outputs from McPAT may not reflect realistic power values [17]. Therefore, the McPAT outputs have to be calibrated to reflect the TDP of the Intel i7 6950× chip. We show the unscaled power values directly collected from McPAT in Figure 3. The chip's total power with ten cores can go to nearly 350 W, which is unrealistic for a ten-core desktop processor. To calibrate the power, we scale all of the steady-state power profiles using the same scaling factor discussed in the previous section, such that the resulting maximum total power matches the chip's TDP, in this case, 140 W as shown in Figure 4.

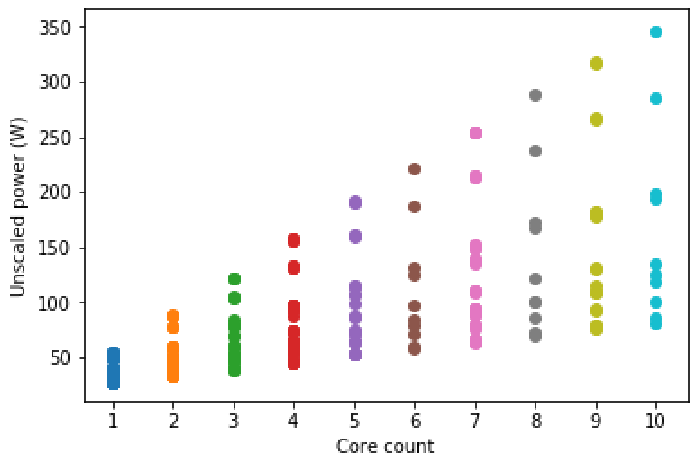

FIGURE 3: ORIGINAL TOTAL CHIP POWER FROM MCPAT, SPLIT BY THE NUMBER OF ENABLED CORES. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED BENCHMARK APPLICATION AND WORKLOAD-CORE MAPPING.

To analyze the total power of each application, we average the total powers over the number of enabled cores and split them by application as shown in Figure 5. We observe that most of the NAS parallel benchmark applications result in an average power

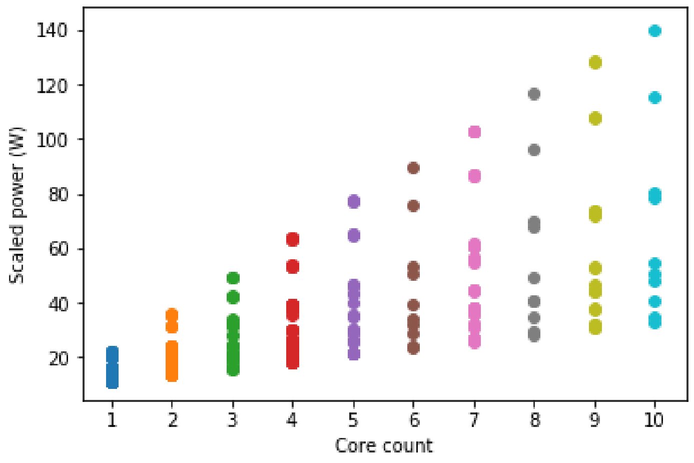

FIGURE 4: SCALED TOTAL CHIP POWER, SPLIT BY THE NUMBER OF ENABLED CORES. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED BENCHMARK APPLICATION AND WORKLOAD-CORE MAPPING.

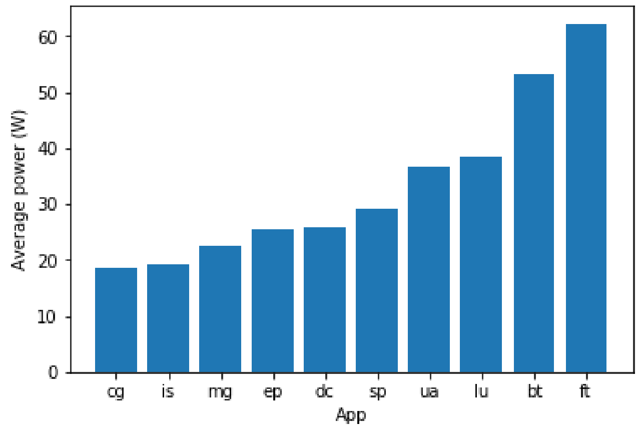

FIGURE 5: AVERAGE POWER OF EACH APPLICATION IN THE NAS PARALLEL BENCHMARKS.

range of 20-40 W. Whereas bt and ft are high power applications and result in high average power of more than 50 W. Applications cg and is are relatively low power applications.

Next, we show the temperature results obtained by running PACT with all the steady-state training power profiles. In Figure 6, we show the average temperatures for applications and split them by the number of enabled cores. Note that we run PACT with the ambient temperature set to 45°C. The average temperature increases as we increase the number of cores to run the application. This is because of the total power increase as we increase the number of enabled cores, as shown in Figure 4. We also show the average temperature for the number of enabled cores to run the applications in Figure 7. Applications bt and ft result in the highest average temperatures, and applications cg and is have the lowest average temperatures. The average temperatures generally follow the trend of their average power behaviors, as shown in Figure 5. However, since other factors such as the hot spots' locations and power densities also affect the average temperature, higher average powers may result in lower average temperatures (e.g., applications dc and lu).

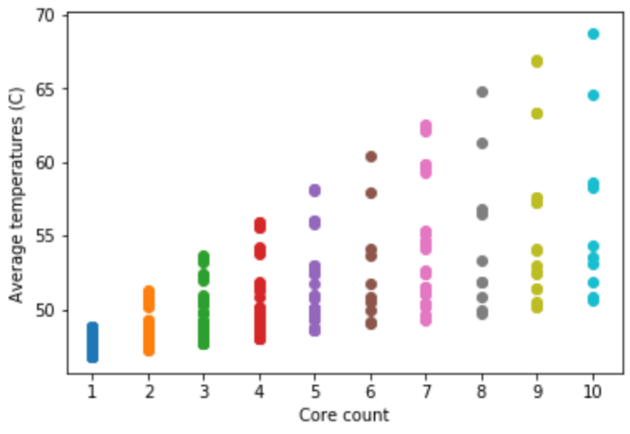

FIGURE 6: AVERAGE TEMPERATURES FOR APPLICATIONS (°C), SPLIT BY THE NUMBER OF ENABLED CORES USED. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED CORE DISTRIBUTION AND BENCHMARK APPLICATION.

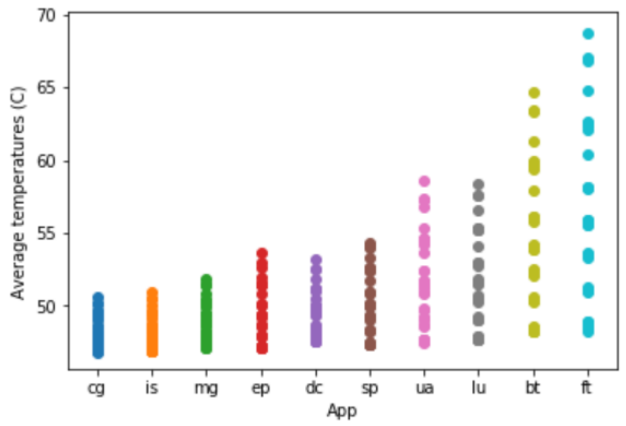

FIGURE 7: AVERAGE TEMPERATURES (°C) FOR THE NUMBER OF ENABLED CORES TO RUN THE APPLICATIONS, SPLIT BY APPLICATION AND SORTED IN ASCENDING ORDER BY AVERAGE POWER. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED NUMBER OF ENABLED CORES AND WORKLOAD-CORE MAPPING.

3.0.2 Validation of the Machine Learning Model. Since our machine learning model uses a random sample of ten temperatures from the temperature grid matrix as the temperature values reported by on-chip thermal sensors (input to the model), we need to show that the model's accuracy is not affected by the randomness of the thermal sensor placements. To test the effect of the randomly selected thermal sensor locations on the model's accuracy, we evaluate the model's R2 score and RMSE across a wide range of random thermal sensor placements. For each random sample of ten thermal sensor readings, we split the data into training and testing data using a 80/20 train-test ratio with the same random state. Using Scikit Learn, we achieve this with the train_test_split function with the random state set to 144. We show the histograms of R2 score and RMSE in Figures 8 and 9. As a result, we observe that the locations of the thermal sensors on the chip can affect the model' accuracy by at most $0.12°C$. This accuracy loss indicates that our proposed machine learning model's accuracy is invariant to the placements of the thermal sensors. We use the thermal sensor placement corresponding to the median error for the remaining model validation results as shown in Figure 10.

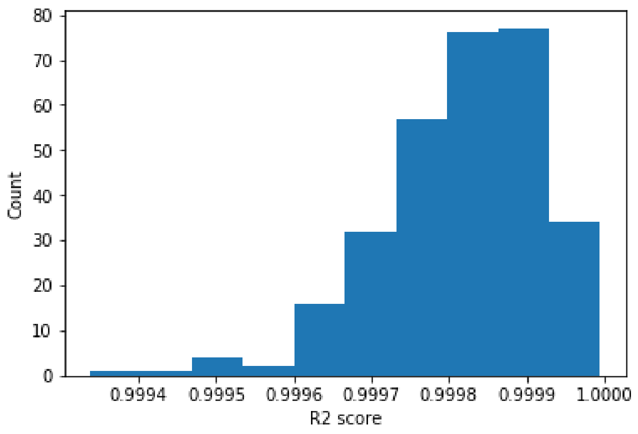

FIGURE 8: HISTOGRAM OF R2 SCORE DISTRIBUTION ACROSS 300 DIFFERENT THERMAL SENSOR PLACEMENTS USED AS INPUT TO THE MODEL.

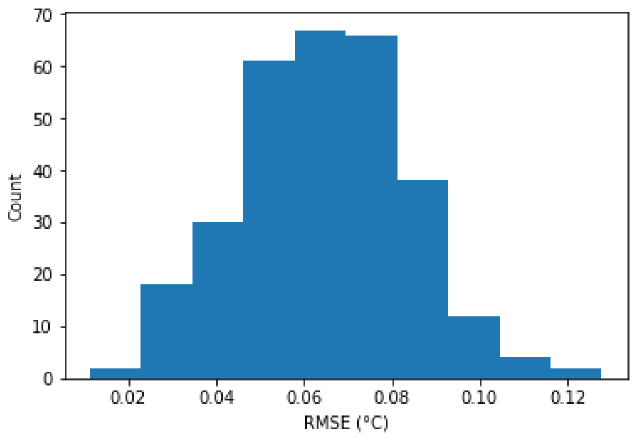

FIGURE 9: HISTOGRAM OF RMSE DISTRIBUTION ACROSS 300 DIFFERENT THERMAL SENSOR PLACEMENTS USED AS INPUT TO THE MODEL.

The results of the 5-fold CV with the selected thermal sensors locations are shown in Table 1. The 5-fold CV results show that the proposed linear regression model has a high coefficient of determination with the lowest R2 score of 0.9996. In addition, the model itself is accurate with the highest RMSE of $0.0695°C$, and a maximum absolute error of $1.3948°C$. We demonstrate the comparison of the golden heat map and the predicted heat map using the machine learning model for the worst-case accuracy of

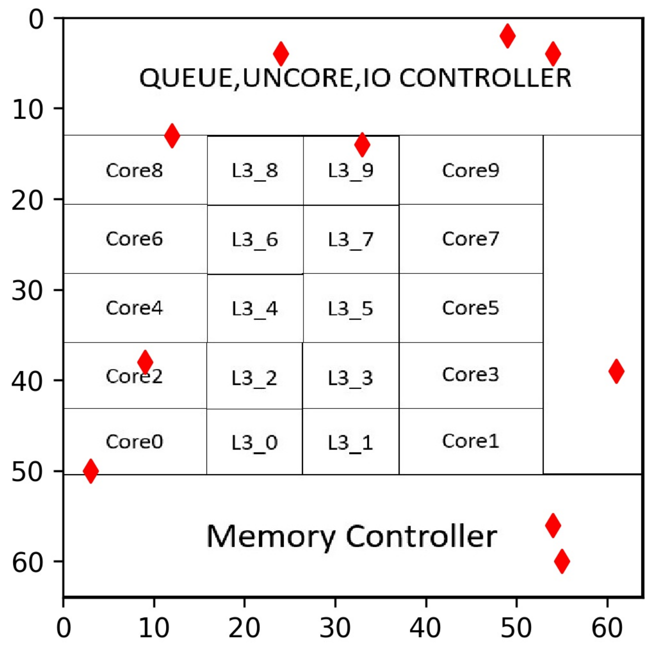

FIGURE 10: THE SELECTED THERMAL SENSOR PLACEMENT. RED MARKERS INDICATE THERMAL SENSORS.

the 5-fold CV experiments in Figure 11. We can observe that the hot spot on the predicted heat map has a lower temperature than the hot spot on the golden heat map. The application that results in the highest accuracy loss, in this case, is ft. There are two reasons behind this accuracy loss. First, application ft is the highest power application, with an average power of more than 60 W. The majority of the applications we used to train the machine learning regression model have average power within the range of 20-40 W, which means our model is more likely to learn the power and thermal trends of these medium power applications. Second, as shown in Figure 4, when the number of enabled cores to run the application is one, the total power of the chip is less than 30 W. Therefore, predicting the heat map for the highest power application and lowest power workload-core mappings results in the highest validation accuracy.

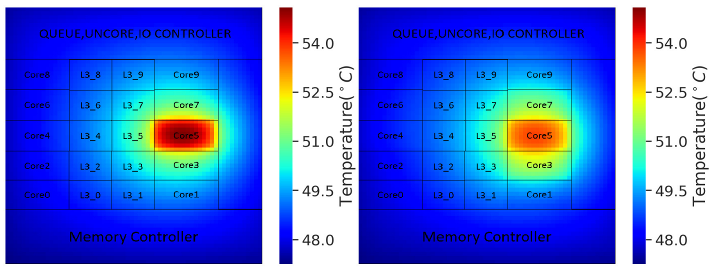

FIGURE 11: HEAT MAP COMPARISON FOR THE WORST CASE. THE LEFT HEAT MAP IS THE GOLDEN HEAT MAP, AND THE RIGHT HEAT MAP IS THE PREDICTED HEAT MAP.

Lastly, to validate that the model training methodology can be applied to various use cases and configurations, we perform LOOCV on both the benchmark applications and the number of enabled cores to run the applications. For the applications, we

TABLE 1: 5-FOLD CROSS-VALIDATION RESULTS.

5-fold CV	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
R2 Score	0.9997	0.9996	0.9997	0.9997	0.9997
RMSE	0.0678	0.0695	0.0657	0.0669	0.0688
Max Absolute Error	1.0438	1.3948	0.9891	1.0464	0.8308

split the data into ten buckets based on the type of the benchmark. Then we train the model using nine buckets, leaving one out for validation. This CV is repeated for each bucket, measuring the model's accuracy with respect to each application. The LOOCV for the number of enabled cores is similar. We split the data based on the number of enabled cores (core counts) to run the application. Then, we train the model for each core using the data containing the rest of the core counts, then test the accuracy on the left out one. We show the LOOCV results on applications in Figures 12 and 13. The comparison of the golden heat map and the predicted heat map for the worst case when the number of enabled cores equal to 5 is shown in Figure 14. The application that causes the highest accuracy loss for five cores is ft. The reason is that most of the applications we used to train the machine learning model are medium power applications (average power within 20-40 W). However, suppose we train the model without high power applications (e.g., bt or ft). In that case, our model predicts high power applications heat maps less accurately and results in an RMSE error of less than 0.25°C, and a maximum absolute error of 1.2°C.

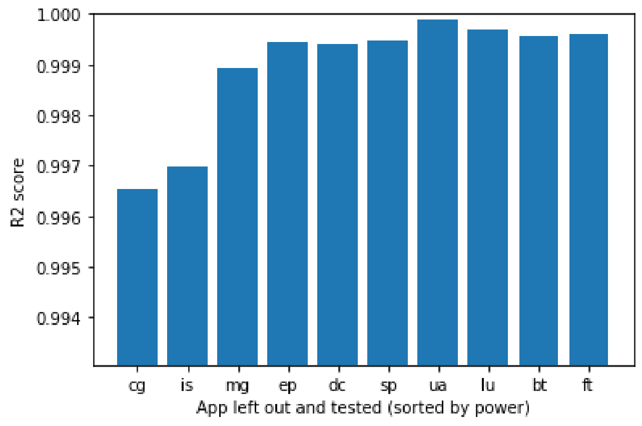

FIGURE 12: AVERAGE LOOCV R2 SCORES ON APPLICATIONS.

In addition, for lower power applications cg and is, the RMSEs are also relatively high. Meanwhile, the applications mg, ep, dc, sp, ua, and lu have lower errors. This trend indicates that this model is generally able to predict CPU temperatures for application workloads not seen in the training data. However, the training data should include the upper and lower extreme applications in terms of power and temperature to get the best accuracy.

We perform similar LOOCV on the number of enabled cores to run the applications. This time, we split the data into buckets based on the number of enabled cores and train the model using all but one of these buckets. We demonstrate the LOOCV on the number of enabled cores in Figures 15 and 16. We also illustrate

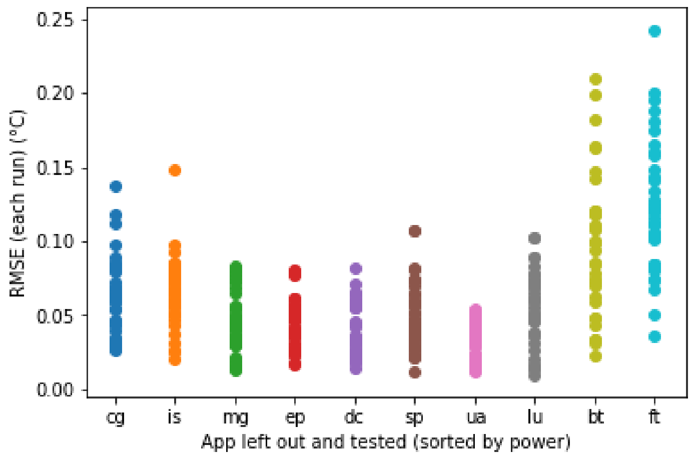

FIGURE 13: LOOCV RMSES ON APPLICATIONS. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED NUMBER OF ENABLED CORES AND WORKLOAD-CORE MAPPING.

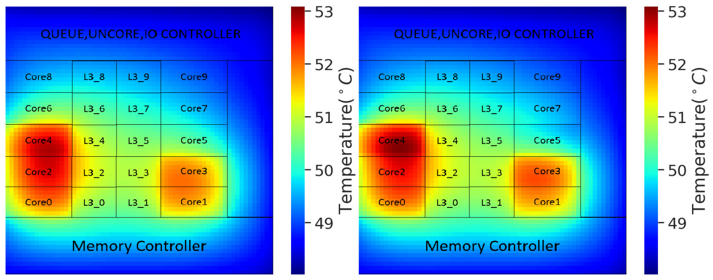

FIGURE 14: HEAT MAP COMPARISON FOR THE WORST CASE WHEN THE NUMBER OF ENABLED CORES IS EQUAL TO 5. THE LEFT HEAT MAP IS THE GOLDEN HEAT MAP, AND THE RIGHT HEAT MAP IS THE PREDICTED HEAT MAP.

the comparison of the golden heat map and the predicted heat map for the worst case when the number of enabled cores equal to ten in Figure 17. Note that the CPU cores in the right column are 1-2°C hotter than the cores in the left column. This is because the cores in the right column are close to the center of the chip compared to the left, which result in a higher temperature hot spot. The application that results in the highest RMSE for ten cores case is lu. The LOOCV on the number of enabled cores results in a maximum absolute error of less than 1.14°C.

As a result, we observe that the overall error of the model does not vary significantly in terms of the number of enabled cores, with a maximum RMSE of less than 0.25°C. The chip's temperature map depends not only on the number of cores that runs the applications but also on the workload-core mapping policies. In this case, different workload-core mapping polices affect the overall temperature, and hence the accuracy variation result from the number of enabled cores is not significant. When training the proposed model, including the full range of workloads would ensure good accuracy when predicting temperatures. The above 5-fold CV and LOOCV results confirm the accuracy and practicality of the machine learning regression models. In addition, our proposed method can take actual temperature data either from actual on-chip thermal sensors readings or IR camera

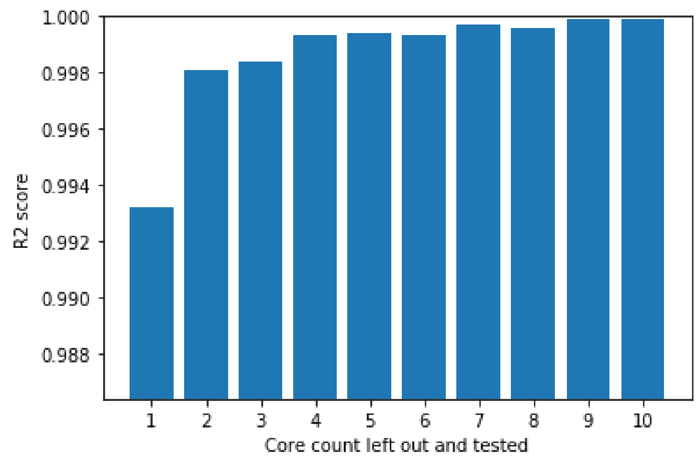

FIGURE 15: AVERAGE LOOCV R2 SCORES ON ENABLED CORES.

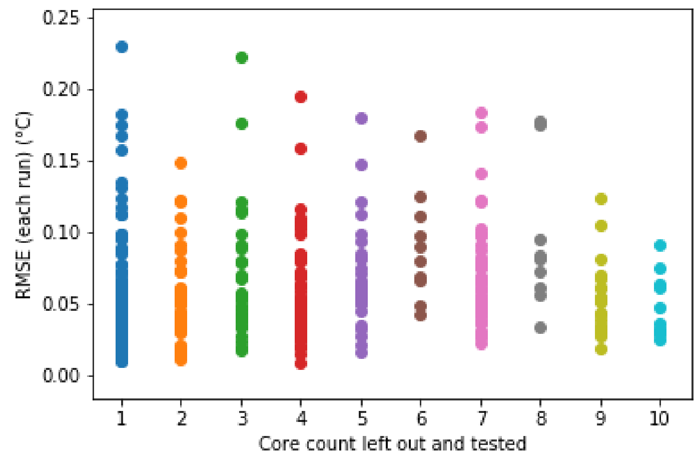

FIGURE 16: LOOCV RMSES ON ENABLED CORES. EACH POINT REPRESENTS AN INDIVIDUAL RUN, WITH A SPECIFIED BENCHMARK APPLICATION AND WORKLOAD-CORE MAPPING.

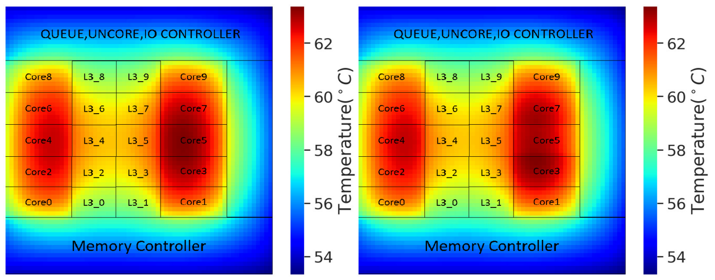

FIGURE 17: HEAT MAP COMPARISON FOR THE WORST CASE WHEN THE NUMBER OF ENABLED CORES IS EQUAL TO 10. THE LEFT HEAT MAP IS THE GOLDEN HEAT MAP, AND THE RIGHT HEAT MAP IS THE PREDICTED HEAT MAP.

measures as a replacement for simulation-based training data.

4. LIMITATIONS AND FINAL REMARKS

This paper introduces a machine learning and simulation-based method to generate a full temperature map based on the total chip power, on-chip thermal sensors measurements, and workload-core mappings. Compared to existing work, the proposed method is low-cost and accurate. We demonstrate that the placements of the thermal sensors do not affect the accuracy of the proposed method and machine learning model. 5-fold CV results prove that the RMSE of the machine learning linear regression model is less than $0.07°C$. We also perform LOOCV on the applications and core mappings. Results confirm that the machine learning model is able to accurately predict the temperature profile of previously unseen applications and the number of enabled cores with a maximum RMSE of less than $0.25°C$.

Note that, in this paper, we select the linear regression model as our machine learning model to predict the temperature maps because it results in good prediction accuracy (a maximum RMSE of less than $0.25°C$) and low simulation time overhead of less than $50\mu s$. For chip stacks with emerging integration (e.g., die-stack 3D or monolithic 3D) and cooling technologies (e.g., liquid cooling and two-phase cooling), a more complex machine learning regression model can be used to replace the current linear regression model to achieve better prediction accuracy if needed.

We identify several limitations of our work as follows:

- First, we only consider a single highly parallel workload running on the chip at a time, which may limit the power variations among CPU cores. In our future work, we will collect power data of running different applications on the chip at the same time and add these data to the training and testing datasets.
- Second, the current methodology of predicting temperature profiles using the machine learning model only supports the prediction of steady-state heat maps. In contrast, in reality, the control knobs of runtime thermal control policies such as thermally-aware dynamic voltage frequency scaling often rely on the instantaneous temperature readings of the thermal sensors. Therefore, the proposed method needs to be extended to support transient heat map prediction.
- Third, we assume that the number of thermal sensors integrated into the Intel i7 6950× equals 10. In our future work, we plan to investigate the number and locations of the thermal sensors on-chip that achieve more accurate temperature readings and the best cost-effectiveness.

ACKNOWLEDGEMENT

This paper has been partially funded by the NSF CRI (CI-NEW) grant #1730316/1730003/1730389, the NSF CCF grant #1910075/1909027, and Boston University's Undergraduate Research Opportunities Program.

REFERENCES

[1] Srinivasan, Jayanth, Adve, Sarita V, Bose, Pradip, Rivers, Jude and Hu, Chao-Kun. "Ramp: A model for reliability aware microprocessor design." *IBM research report* .

[2] Saini, Pushpa and Mehra, Rajesh. "Leakage power reduction in CMOS VLSI circuits." *International Journal of Computer Applications* Vol. 55 No. 8.

[3] Sheikh, Hafiz Fahad, Ahmad, Ishfaq, Wang, Zhe and Ranka, Sanjay. "An overview and classification of thermal-aware scheduling techniques for multi-core processing systems." *Sustainable Computing: Informatics and Systems* Vol. 2 No. 3 (2012): pp. 151–169.

[4] Reda, Sherief, Cochran, Ryan and Nowroz, Abdullah Nazma. "Improved thermal tracking for processors using hard and soft sensor allocation techniques." *IEEE Transactions on Computers* Vol. 60 No. 6 (2011): pp. 841–851.

[5] Sadiqbatcha, Sheriff I, Zhang, Jinwei, Amrouch, Hussam and Tan, Sheldon X-D. "Real-Time Full-Chip Thermal Tracking: A Post-Silicon, Machine Learning Perspective." *IEEE Transactions on Computers* .

[6] Sharifi, Shervin and Rosing, Tajana Šimunić. "Accurate Direct and Indirect On-Chip Temperature Sensing for Efficient Dynamic Thermal Management." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* Vol. 29 No. 10 (2010): pp. 1586–1599. DOI 10.1109/TCAD.2010.2061310.

[7] Long, Jieyi, Memik, Seda Ogrenci, Memik, Gokhan and Mukherjee, Rajarshi. "Thermal Monitoring Mechanisms for Chip Multiprocessors." *ACM Trans. Archit. Code Optim.* Vol. 5 No. 2. DOI 10.1145/1400112.1400114. URL https://doi.org/10.1145/1400112.1400114.

[8] Zhang, Jinwei, Sadiqbatcha, Sheriff, O'Dea, Michael, Amrouch, Hussam and Tan, Sheldon X-D. "Full-Chip Power Density and Thermal Map Characterization for Commercial Microprocessors under Heat Sink Cooling." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* .

[9] Nowroz, Abdullah Nazma, Cochran, Ryan and Reda, Sherief. "Thermal monitoring of real processors: Techniques for sensor allocation and full characterization." *Design Automation Conference*: pp. 56–61. 2010. IEEE.

[10] Chrobak, Marek, Dürr, Christoph, Hurand, Mathilde and Robert, Julien. "Algorithms for temperature-aware task scheduling in microprocessor systems." *International Conference on Algorithmic Applications in Management*: pp. 120–130. 2008. Springer.

[11] Sima, Dezső. "Intel Core X-series (HED lines)." .

[12] Bailey, David H et al. "The NAS parallel benchmarks." *International Journal of Supercomputing Applications* Vol. 5 No. 3 (1991): pp. 63–73.

[13] Carlson, Trevor E, Heirman, Wim and Eeckhout, Lieven. "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation." *ACM Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*: p. 52. 2011.

[14] Li, Sheng et al. "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures." *IEEE/ACM Proc. of 42nd Annual International Symposium on Microarchitecture (MICRO)*: pp. 469–480. 2009.

[15] Yuan, Zihao, Shukla, Prachi, Chetoui, Sofiane, Nemtzow, Sean, Reda, Sherief and Coskun, Ayse K. "PACT: An Extensible Parallel Thermal Simulator for Emerging Integration and Cooling Technologies." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* Vol. 41 No. 4 (2022): pp. 1048–1061. DOI 10.1109/TCAD.2021.3079166.

[16] Yuan, Zihao, Zhang, Tao, Van Duren, Jeroen and Coskun, Ayse K. "Efficient Thermal Analysis of Lab-Grown Diamond Heat Spreaders." *International Electronic Packaging Technical Conference and Exhibition*, Vol. 85505: p. V001T01A002. 2021. American Society of Mechanical Engineers.

[17] Lee, Wooseok, Kim, Youngchun, Ryoo, Jee Ho, Sunwoo, Dam, Gerstlauer, Andreas and John, Lizy K. "PowerTrain: A learning-based calibration of McPAT power models." *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*: pp. 189–194. 2015. IEEE.