Steady-State Temperature Prediction Based on Compact Thermal Models Using Machine Learning

Mohammadamin Hajikhodaverdian*, Sherief Reda†, Ayse K. Coskun*
* Boston University - (aminhaji, acoskun)@bu.edu
† Brown University - sherief_reda@brown.edu

***Abstract*—With the scaling up of transistor densities, the thermal management of integrated circuits (IC) in 3D designs is becoming challenging. Conventional simulation methods, such as finite element methods, are accurate but computationally expensive. Compact thermal models (CTMs) provide an effective alternative and produce accurate thermal simulations using numerical solvers. Recent work has also designed machine learning (ML) models for predicting thermal maps. However, most of these ML models are limited by the need for a large dataset to train and a long training time for large chip designs.**

To overcome these challenges, we present a novel ML framework that integrates with CTMs to accelerate thermal simulations without the need for large datasets. We introduce a methodology that effectively combines the accuracy of CTMs with the efficiency of ML using a physically informed linear regression model based on the thermal conduction equation. We further introduce a window-based model reduction technique for scalability across a range of grid sizes and system architectures by reducing computational overhead without sacrificing accuracy. Unlike most of the existing ML methods for temperature prediction, our model adapts to changes in floorplans and architectures with minimum retraining. Experimental results show that our method achieves up to $70\times$ speedup over the state-of-the-art thermal simulators and enables real-time, high-resolution thermal simulations on different IC designs from 2D to 3D. [1]

***Index Terms*—Thermal simulation, Compact thermal models (CTM), Machine learning, 3D IC**

I. Introduction

The increasing transistor densities in chips lead to higher power and heat densities, which can degrade chip performance and negatively impact the reliability and lifespan of semiconductor devices [1]–[3]. To overcome the limitations of 2D scaling in width and length, 3D design architectures have emerged through modular architectures with multiple stacks that deliver high performance, increased bandwidth, and reduced area costs. However, these advantages come at the cost of notably increased heat densities due to several active silicon layers on top of each other, further increasing the thermal handling issue that already arises with higher transistor densities [4]. Therefore, thermal analysis during chip design becomes a necessary and unavoidable task. Conventional methods for creating a thermal map of a chip require solving the heat conduction equations using finite element methods (FEM) [5], which are utilized in commercial software, such as COMSOL and ANSYS. These approaches necessitate considerable computational time and memory resources. In order to mitigate the computational time and memory constraints associated with FEM-based techniques, researchers have developed compact thermal models (CTMs) that facilitate accelerated thermal simulation while maintaining promising levels of accuracy [6]–[8]. CTMs simplify the thermal problem by constructing lumped thermal resistance and capacitance networks. CTMs are widely used for temperature simulation and even extended to account for advanced cooling methods, e.g., microchannels, heat sinks, and hybrid cooling [9], [10]. However, for temperature simulation of a specific design, the user has different workflows and power maps that must run CTMs a few hundred or even thousands of times. For each new power map, the numerical solver of the CTMs needs to run several iterations to converge, resulting in a long simulation time. Recent successes in machine learning (ML)-based methods promise accurate temperature predictions. Various works have investigated ML-based approaches for predicting on-chip temperature distribution [11]–[13]. Unfortunately, most such techniques require large datasets for effective execution and are computationally intensive during training. They often need retraining as new architectural designs need to be simulated. Moreover, some of these models rely on supportive inputs from temperature information derived through infrared (IR) sensors [14] or simulations [15], which is expensive and impractical for widespread application.

This work, which extends our preliminary work [16], presents a new ML method based on CTMs that can train the model with a few samples, reduce the training time for each design, and bypass the CTMs when they reach a reasonable accuracy. Most of the prior works using ML models did not consider the physics of the problem, and they designed complicated models for steady-state simulation, which is a linear problem solved in CTMs. To the best of our knowledge, this is the first work introducing a framework that can speed up a CTM simulation time by utilizing a fast and reliable ML model. Our contributions are summarized below:

- High-resolution thermal modeling using ML needs many training samples for reasonably accurate temperature prediction. We address this problem by reducing needed training samples to only a few (less than 10 for grid sizes of 64×64 and 128×128) while achieving high accuracy, using a model that considers the physics of the problem.
- We design a low-latency framework that is needed in architecture optimization and design, where temperature simulations are often performed on complex 3D inte-

[1]This research was partially funded by the NSF CCF 2131127 grant

grated circuit (IC) geometries and designs with multiple variations in their floorplans.
- We introduce a model reduction approach based on a windowing methodology, enhancing scalability for a wide range of grid dimensions and system architectures, hence offering an effective solution to the computational problems connected with high-resolution thermal simulations.

As part of our evaluation, we compared our implemented method against an existing compact thermal simulator (CTS) and demonstrated a significant speedup of more than $70\times$ for complicated designs. This result highlights the potential for fast, real-time thermal simulations while maintaining accuracy, showcasing the practical benefits of our approach. The subsequent sections of this paper are organized as follows: Section II reviews the related literature. Section III details our designed methods and presents a performance evaluation of it in Section IV. Finally, our conclusion is drawn in Section V.

II. Related Work

Challenges existing in CTMs and conventional FEM-based approaches have been addressed by the emergence of several ML-based methods as a promising alternative for thermal simulation. The ML-based thermal analysis runs faster compared with traditional numerical solvers while still maintaining high accuracy. There are examples leveraging ML models training on IR camera measurements of physical chips [14], [17]. While such methods have offered highly accurate temperature prediction, their wider adoption within the research community is restricted due to the high expense of IR camera equipment. Moreover, they can only apply to post-silicon designs.

On the other hand, several recent works have developed simulation-based methods using ML models such as graph neural networks and CNN-based encoder-decoder architectures [18]–[21] that avoid the use of costly IR camera setups. These techniques provide accurate and efficient thermal predictions during the pre-silicon design stages. Thus, they are more feasible for a wide range of applications. However, they rely on complex ML models to predict temperature, even though steady-state temperature prediction (with uniform material properties) can often be modeled by a linear equation, as in the case of this paper. For more general temperature modeling (transient, nonuniform materials, etc.), nonlinear effects need to be considered. These models need large datasets to train all the parameters, increasing the training time and slowing down the temperature modeling. The extensive data required for training are usually generated through time-consuming and computationally expensive operations, such as the thousands of runs of CTMs or FEM-based simulations. Another disadvantage is that significant changes in the design floorplan require retraining, further exacerbating the data collection problem.

For instance, Ranade et al. [19] had to train different models for various use cases, where each model required thousands of samples; Chhabria et al. [21] utilized a convolutional neural network with downsampling and upsampling, which required the retraining of the model with thousands of samples for every new case. Wen et al. [20] predicted temperature differences and afterward used coarse-grain FEM simulations to compute the final temperature maps.

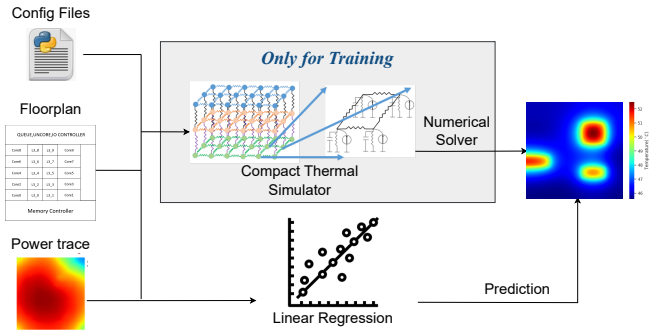


Fig. 1: Framework of our method: Trained on power trace data and corresponding temperature maps from the compact thermal simulator, the ML model then forecasts temperatures directly, eliminating the need for additional simulation.

Unlike previous methods, which require large datasets and retraining whenever applications change, or new floorplan designs are devised, our method implicitly embeds the physical laws driving the thermal characteristics. As such, it allows for more robust generalization with minimal retraining if changes in the design are made. Unlike some of the previous works that rely on additional coarse-grain FEM simulations after prediction, our method will avoid the necessity of extra simulation by directly providing accurate temperature predictions. Moreover, our window-based model drastically decreases the number of training parameters, reducing computational requirements and allowing it to run on low-performance hardware. The window-based model can also increase the scalability of our design, which previous works did not consider fine-grain simulations. This broadly applicable, operationally efficient integration with minimum data requirements improves the prior work and effectively solves the challenges related to extensive data preparation, retraining, and higher computation costs.

III. Predicting Steady-State Temperature: A CTM-based Machine Learning Approach

In this work, we design a method to accelerate the process of compact thermal simulations using an ML model. Our approach addresses the computational limitations of traditional thermal solvers and offers a scalable and efficient alternative that aligns with the physics of thermal conduction in IC designs. First, we summarize key physical aspects in CTM that guided us in choosing our model, followed by a discussion of our CTM-based design. Finally, we describe the window-based model that we use.

A. Compact Thermal Model

The CTM approach creates thermal circuits based on the duality between thermal and electric fields, allowing reasonably accurate temperature predictions [6], [8]. To model a chiplet-based system, CTM divides each layer into cuboid grids. Based on these grids, equivalent thermal resistance networks

are constructed. Each grid cell is represented as a node, and nodes are connected to their neighboring cells. The linear matrix equation then describes the system:

$$GT = P \tag{1}$$

where T is the vector of node temperatures (K), G is the conductance matrix (W/K), and P represents the power consumption at each node (W). Compact thermal simulators (CTS) apply CTM for temperature predictions and they mainly use numerical solvers to solve linear equations in Eq. (1). For a chiplet, the matrix G is high-dimensional and these numerical solvers are inefficient and time-consuming in those cases.

B. Model Design

Our introduced approach combines an ML model with a CTS to speed up the temperature prediction in the IC designs. Instead of solving the linear systems in Eq. (1) for any new P using a numerical solver as conventional CTS do, we design a method to estimate G^{-1} using ML models; since $T = G^{-1}P$, by having G^{-1}, computing temperature for any arbitrarily P would become straightforward. Although computing temperature is straightforward once G^{-1} is available, explicitly inverting G can be expensive for large-scale grids (e.g., 256×256 or 3D designs). This is because the matrix size grows quadratically with increasing the resolution, making explicit calculation time-consuming and memory intensive. In contrast, our ML-based approach also involves an upfront cost but is far more flexible when the floorplan changes frequently. Rather than recomputing the full matrix inversion for each new layout, our method only needs a partial update of the learned model, significantly reducing overhead.

Since the nature of thermal conduction in steady-state CTMs and the Eq. (1) are linear, we select linear regression as our base ML model to estimate the inverse thermal conductance matrix. Not only linear regression aligns with the physics of the problem, but it is also a simple and fast model to train. Unlike previous work, we demonstrate that our model is a lightweight model compared to other complicated models. Our model requires no more than a few samples to achieve high accuracy (less than 10 samples for grid sizes of 64×64 and 128×128). Although a 64×64 grid causes thousands of parameters to be trained, most of these values in the matrix G are zero. This makes it simple for the linear regression model to learn important values faster with less data. In addition, since the conduction matrix is considered uniform based on the uniform material properties we considered in this paper, the heat conduction follows a linear pattern, which simplifies the learning process of our model.

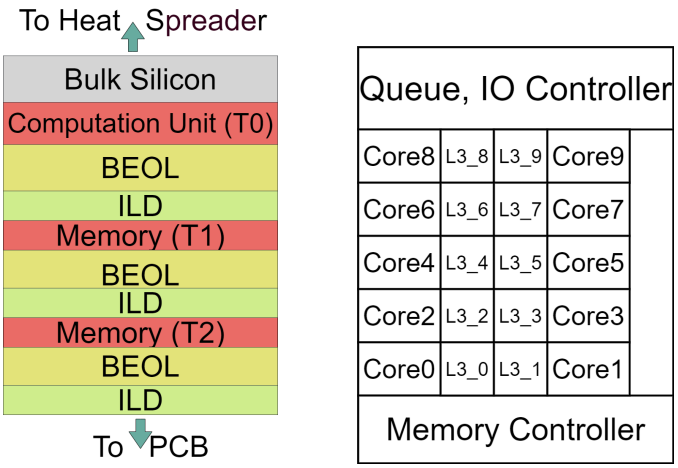
In our workflow, shown in Fig.1, the input data is represented by configuration files (defining grid size, simulation time, etc), floorplans, and power traces. Our approach is separated into a training phase and an inference phase. In the training phase, we use CTS to solve the thermal conduction equation (Eq. (1)) using a numerical solver to obtain an accurate temperature map. The training samples are selected randomly since the critical information (G matrix) we need from these power maps is identical if the samples are from the same architecture. We feed training power maps to the ML model and predict the temperature. Model parameters in this method will be updated by considering the whole grid size resolution. To switch to our inference phase, we set a training threshold based on either our error metrics (e.g., mean square error (MSE)) or when the model converges to a specific error, meaning it has learned adequately the relation between power and temperature, to stop using CTS. After achieving this threshold, the model is allowed to bypass the numerical solver within the CTS framework and generate temperature predictions using the ML model, hence drastically accelerating the process of the simulation. The main advantage of the ML model is the capability to predict temperatures quickly and without deep numerical simulations within CTS performed by the numerical solver.

Algorithm 1 Window-based Linear Regression for Temperature Prediction

Input: Power matrix P of size $N \times N$, Temperature matrix T, Window size w, Stride s
Output: Predicted temperature matrix T_{pred}
Initialize $T_{pred\_acc} \leftarrow 0$ of size $N \times N$
Initialize $Count \leftarrow 0$ of size $N \times N$
for $i = 0$ to $N - w$ with step size s **do**
  for $j = 0$ to $N - w$ with step size s **do**
    $Power\_window \leftarrow P[i : i + w, j : j + w]$
    $Temp\_window \leftarrow T[i : i + w, j : j + w]$
    $Temp\_pred \leftarrow$ Linear-Regression($Power\_window$)
    $T_{pred\_acc}[i : i+w, j : j+w] \leftarrow T_{pred\_acc}[i : i+w, j : j + w] + Temp\_pred$
    $Count[i : i + w, j : j + w] \leftarrow Count[i : i + w, j : j + w] + 1$
  end for
end for
Average nodes with several prediction:$T_{pred} \leftarrow \frac{T_{pred\_acc}}{Count}$
Return: T_{pred}

C. Window-based Linear Regression

While the linear regression approach is straightforward, using a large grid (e.g., 256×256 or 512×512) results in a polynomial increase in the number of model parameters, which can lead to memory limitations and higher computational demands. To address these challenges in high-resolution simulations, we introduce a windowing technique that partitions the grid into smaller sections (e.g., 32×32 or 64×64). This approach is effective due to the typically uniform nature of the conductance matrix across the design. The calculation of G^{-1} on a small grid results in similar accuracy as for the full chip since the values are uniform. Our window-based method is described in Algorithm.1. In our algorithm, first, we divide our power and temperature map into smaller windows based on the window size and stride step to have overlapping windows. We then apply linear regression to each window, adjusting the model's parameters based on the error in each window sample. By sweeping overlapping windows across the grid, we capture more accurate power propagation



(a) Monolithic 3D architecture (b) 2D floorplan

Fig. 2: (a) A monolithic 3D architecture with three active layers, one computation unit layer, and two memory layers, (b) Top view of block-level floorplan that estimates Intel i7 as a target

between neighboring regions. Then, we aggregate all of the predictions. By tracking the number of predictions for each node ($Count$ matrix), we compute the average of multiple temperature estimates for overlapping nodes to ensure the predictions remain within a reasonable range. Choosing an appropriate window size is crucial since selecting a window that is too small compared with the grid can prolong simulation time and decrease the accuracy of the prediction. Generally, we observed a window size that is about a quarter of the grid provides a good balance between low model complexity while maintaining high accuracy and speed.

IV. Evaluation

This section first describes our method's experimental setup and then demonstrates our ML model's benefits. Our primary objective is to evaluate the accuracy of our framework across various simulation resolutions and methods by comparing its outputs with actual results generated by a compact thermal simulator. We also measure the speedup improvements gained by using this framework.

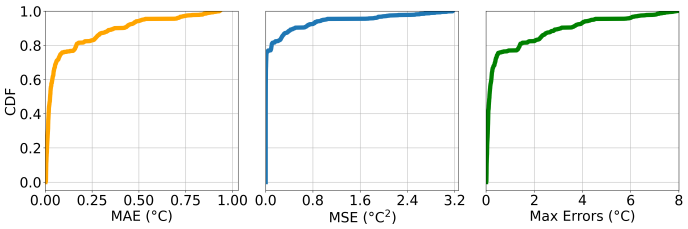
A. Experimental Setup

We evaluate our method on multiple designs. For the 2D design, we use Intel i7 6950× processor block level floorplan, as shown in Fig. 2b, and we run ten different applications from the NAS parallel benchmarks [22] using Sniper [23] and generate power consumption traces with McPAT [24]. We configure the input file in McPAT by instantiating each component (e.g., core0, core1, etc.), ensuring accurate power modeling for all cores and components. This allows us to simulate the power usage and generate the power traces required for thermal simulation in PACT. Maximum total power in this case is $140W$. In our 3D design simulation, we evaluate a monolithic 3D architecture with two configurations: one with three active layers and another with two active layers. In both designs, the total power of each layer does not exceed $140W$. Each layer has a maximum temperature of near $100°C$ to demonstrate that our method can handle real-world scenarios even with high-temperature differences in the chip (Fig.2 shows a top view of 2D and 3D architecture used in our experiments). Additionally, we use a fixed chip size of $20mm \times 20mm$ and double the number of compute cores within this fixed area. As a result, the size of each core was reduced to fit within the same chip dimensions. We consider the whole chip's maximum total power to be $200W$, distributing it to different cores with a uniform distribution. This setup demonstrates how our model can adapt to changes in the floorplan, with only a few samples. For the grid, we used sizes of 64×64, 128×128, 256×256, and 512×512, while in the windowed linear regression experiment, we employed a grid window size of 32×32 and 64×64 windows with a stride step size of 4 and 16, respectively.

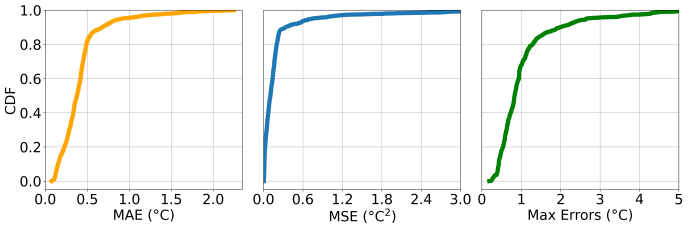
We adopt the compact thermal simulator PACT [8] in our approach. PACT stands out among other simulators due to its parallelized architecture, enabling the use of multiple CPUs to accelerate thermal simulations. Since it is already efficient by nature, we select PACT in this work to demonstrate our approach can further speed up the simulation even when running on top of the state-of-the-art parallel simulator. In our experimental evaluation, the performance of our method will be assessed using a number of key metrics: simulation time as an indicator of computational efficiency, the mean squared error (MSE), and mean absolute error (MAE) of all temperature predictions of the chip for quantifying the accuracy of temperature prediction given ground truth data. Moreover, we considered the peak temperature difference (maximum error) on the whole chip, which is relevant to fixing the performance of our method in terms of detecting hot spots and thermal variations, particularly under steep temperature gradient conditions. These measurements allow for a thorough check of the performance in terms of both efficiency and accuracy of our methodology. To compare the model's prediction time fairly with the PACT simulation, the training of the model was standardized to 16 CPU cores without using any GPU cores. This allows the estimation of the performance of both the model and the PACT simulation within a comparable framework.

B. Results

In our initial experiments, we assess the accuracy of our model on the Intel i7 processor architecture. The simulation is conducted on four different grids, as mentioned in IV-A, comparing two approaches: a standard linear regression model and a windowed linear regression model with 32×32 and 64×64 window sizes. For the standard linear regression model, we require 6 input samples for training for our three different grid sizes (the grid size of 512×512 is inefficient using a standard linear regression because of high memory and computation needed for updating the parameters (Sec.III-C)). In comparison, the windowed version needs 8 to 12 input samples for all of the different grid sizes. After training, we can infer the remaining inputs, showcasing the model's ability to generalize effectively with a few training samples.

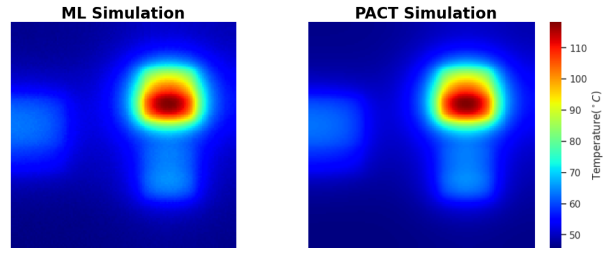


(a) CDF of MSE, MAE, and max temperature error for standard linear regression

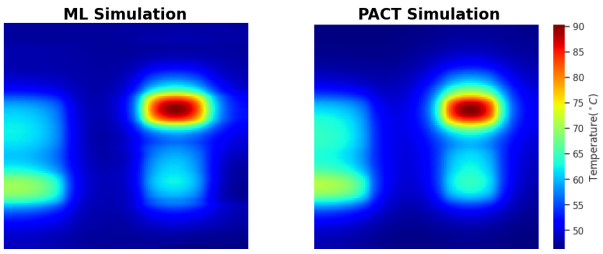


(b) CDF of MSE, MAE, and max temperature error for windowed linear regression

Fig. 3: a) MSE, MAE, and Max temperature error comparison of Intel i7 with a grid size of 64×64, 128×128, and 256×256 for standard linear regression (1056 total samples) b) MSE, MAE, and Max temperature error comparison of Intel i7 with a grid size of 64×64, 128×128, 256×256, and 512×512 for window-based linear regression (1228 total samples)



(a) PACT and ML prediction using standard linear regression



(b) PACT and ML prediction using windowed linear regression

Fig. 4: Thermal map comparison between PACT simulation output and our method prediction

Fig.3 shows the cumulative distribution function (CDF) of the MSE and MAE for inference inputs across both models with different grid sizes. For standard linear regression, more than 90% of the inputs have MSE below $0.5°C^2$. More than 95% of the inputs have MAE below $0.7°C$. These results are comparable to those obtained using the windowed model. However, the windowed approach shows a slower reduction pace due to its fewer parameters compared to standard regression and it needs to go through all of the grids with an

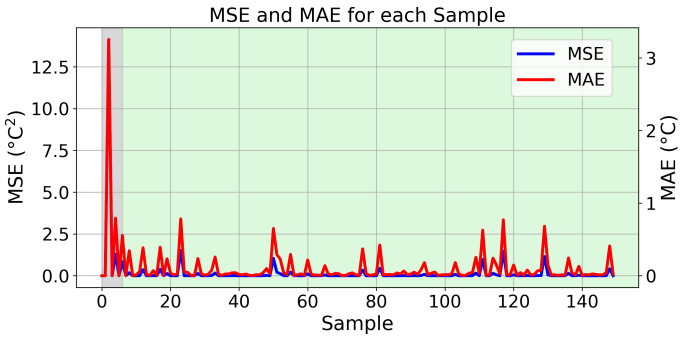


Fig. 5: MSE and MAE error for 3-layer 3D architecture with a grid size of 128×128 over 140 different samples

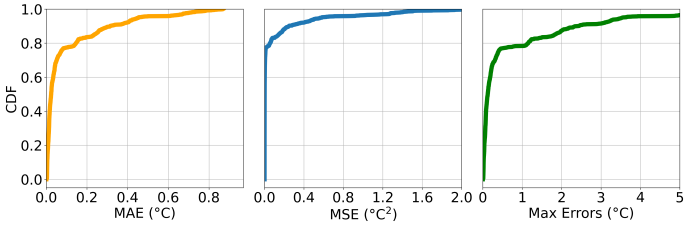


Fig. 6: CDF of MSE, MAE, and Max Temperature Error of 3-layer and 2-layer 3D architecture with grid sizes of 64×64 and 128×128 (1170 total input samples)

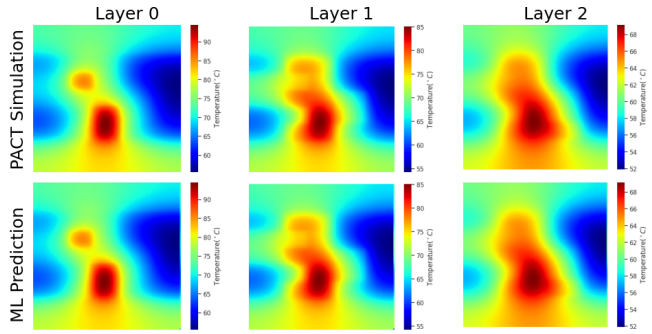


Fig. 7: PACT and ML prediction for each layer of an input sample of monolithic 3D design

overlap step. As seen, both models demonstrate promising inference performance that more than 83% of the inputs have maximum temperature error below $2°C$. With the windowed regression, we reduce the model size by a factor of 256 in grid size of 256×256 while preserving accuracy. Fig.4 compares PACT's outputs with our model's predictions for two sample cases, showing that the general and windowed models produce accurate results. Achieving high accuracy over different grids demonstrates the scalability of our approach while maintaining accuracy at larger grid sizes.

To further validate the robustness of our method in handling complex architectures, we applied it to two monolithic 3D designs (Fig.2a), one with two and the other with three layers, to predict temperature distribution. The simulation utilizes 64×64 and 128×128 grids, ensuring consistency with our earlier experiments. As depicted in Fig.5, as an example, the MSE and MAE were assessed over 140 samples for three-layer 3D architecture. The gray area in the figure corresponds to the

training phase, with only 6 samples used to train the model, while the green area corresponds to the inference phase. The method keeps the error low even with such a complex 3D architecture. It maintains MAE on inference not higher than $2°C$, showing it effectively generalizes with a small number of training samples while keeping prediction quality. Fig.6 presents that for both MSE and MAE, above 95% of the inference samples for all of our simulations on different 3D architectures with various grid sizes are below $0.5°C^2$ and $0.7°C$ respectively. The maximum temperature error, as in Fig.6, above 87% of the inference samples, are below $0.5°C$. That reflects the scalability and adaptability of the method to more complicated architectures. Fig.7 shows the comparison between the output of PACT and the prediction of our method for a particular sample, which demonstrates a strong match. A large temperature gradient and a hot spot near $100°C$ appear in this experiment, showcasing the difficulty of the testing scenario with extreme thermal conditions. Our method predicts the temperature distribution accurately despite the extremes in this experiment, strengthening the robustness of our method to handle sharp gradients and corner cases effectively, even at high temperatures. We also try to simulate with a grid size of 256×256 for a couple of samples from the 2-layer 3D configuration (less than 20 input samples), which, for PACT, takes around 15 to 25 minutes to generate the output. Still, our model can predict the chip temperature after training with a few samples in just 40 to 50 seconds.

In the following set of experiments, we aim to demonstrate the ability of our approach to adapt and fine-tune itself for new architectures and floorplans using only a few training samples. As described in SectionIV-A, we use a fixed chip with random floorplans. We generate 100 unique power maps for each floorplan. Then, after processing these power maps, we doubled the number of computation cores on the chip; thus, the floorplans are increasingly complex. When the floorplan gets complicated, more training samples are needed to maintain high accuracy. We further increase the number of cores to 8 and simulate them with grid sizes of 64×64 and 128×128. For this experiment, we have 800 inputs (400 various power maps for eight different floorplans), and our method only uses 20 of those samples to detect the changes in the floorplan and maintain a high accuracy. The MAE and max error of nearly 95% of the inference samples for all inputs are below $0.5°C$, respectively. This experiment shows that our method can effectively tune itself after training with only several samples and then accurately infer the rest. In cases where more training samples are required, this occurs when there is a significant architectural shift. Such large architecture changes take more data to better align the model for high performance.

We compare our method's simulation time with that of PACT, a state-of-the-art parallel thermal simulator. As shown in Fig.8, we measured the simulation time of the ML model and PACT by summing up the results of experiments that shared the same architecture and simulation setup but had different input power maps, leading to fewer bars than the total number of experiments conducted and compared them. Our method's simulation time accounts for the training and inference stages, as well as the time required to generate the

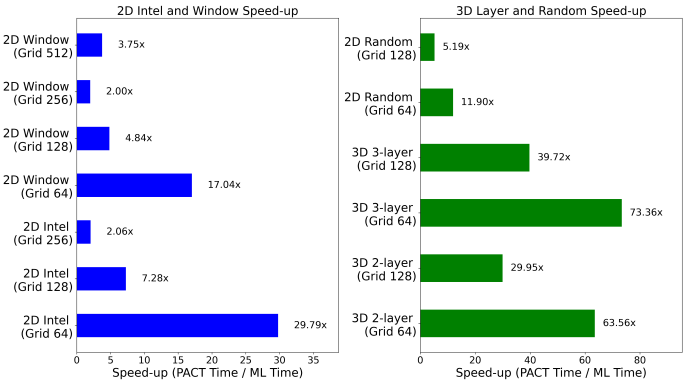


Fig. 8: Comparing the speedup of our method over all different simulations, we consistently improve the simulation time.

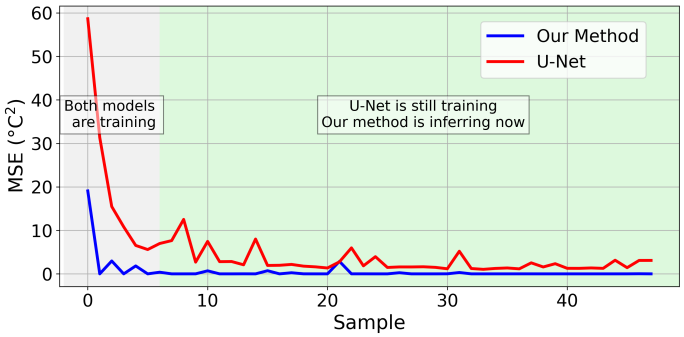


Fig. 9: Comparing our method with U-net model [21] in terms of MSE over different input samples

Model	Mean Time (ms)	Standard Deviation (ms)
U-Net [21]	16.35	6.2×10^{-3}
GCN [18]	38.56	2.02×10^{-3}
Our model	10.74	1×10^{-3}

TABLE I: Comparison of inference time of models with mean time and standard deviation of 1000 runs.

power maps used as inputs for our ML model, following the same approach used for PACT. Our method significantly accelerates the simulation process and achieves a speedup as high as $70\times$, depending on the complexity of the design and the number of input samples. This shows that our method really shines on more complicated architectures (e.g., monolithic 3D architectures) where simulation time is consistently lower than that of PACT while maintaining accuracy at a very high level. The runtime variation seen in the results based on the grid size is because of the need for a few samples generated by PACT which has different time simulations based on the resolution of the simulation and also the number of parameters we use in the ML approach for each grid size prediction. Regarding random floorplans, regular modification in the layout can result in multiple training sessions that will surely slow down our approach. However, this is not a common situation in real applications where frequent retraining would be rarely required. In practice, without this additional overhead, our approach would still obtain very important speedups.

Finally, we compare our method with one of the previous

models that have been utilized for temperature prediction. We use the designed U-net model from [21] instead of our introduced linear regression. Fig.9 compares MSE for our approach and U-Net on unseen data. It demonstrates that our model achieves the intended accuracy within 6 samples; however, U-net needs many more samples for training to achieve similar accuracy. Our method significantly reduces training time and the number of samples needed, from thousands of samples to fewer than 15. We train our model with only a few samples since linear regression in comparison with other methods has fewer parameters, relies on linear relation between inputs and outputs, and can often find the best fit directly using a closed-form formula. As a result, it reaches an acceptable accuracy after training with a few samples. This low-latency framework is especially crucial in architecture optimization and design, where temperature simulations are often performed repeatedly on changing floorplans. Not only have we reduced the training time, but we show in Table I that our model has a comparable inference time to other methods.

V. Summary

This paper introduces a new fast ML-based method for predicting IC steady-state temperatures using CTMs. Traditional approaches like FEM are computationally intensive, requiring significant time and memory for large, complex designs. Our method integrates ML models with CTMs to achieve fast thermal simulations by capturing the underlying physics of the thermal model. Our model requires a few training samples and does not sacrifice accuracy. We use linear regression aligned with thermal conduction principles and a windowed approach to handle larger grid dimensions, reducing computation. As demonstrated, our model's accuracy is promising compared to traditional CTS, which is commonly used as a benchmark against ML-based thermal predictor methods. Tested on 2D and 3D chip designs, our method achieves up to a 70× speedup over simulators like PACT while maintaining high accuracy.

References

[1] Y. Sun, C. Zhan, J. Guo, Y. Fu, G. Li, and J. Xia, "Localized thermal effect of sub-16nm finfet technologies and its impact on circuit reliability designs and methodologies," in *2015 IEEE International Reliability Physics Symposium*, 2015, pp. 3D.2.1–3D.2.6.

[2] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in vlsi circuits: Principles and methods," *Proceedings of the IEEE*, vol. 94, no. 8, pp. 1487–1501, 2006.

[3] S. Makovejev, S. H. Olsen, V. Kilchytska, and J.-P. Raskin, "Time and frequency domain characterization of transistor self-heating," *IEEE Transactions on Electron Devices*, vol. 60, no. 6, pp. 1844–1851, 2013.

[4] K. Cao, J. Zhou, T. Wei, M. Chen, S. Hu, and K. Li, "A survey of optimization techniques for thermal-aware 3d processors," *Journal of Systems Architecture*, vol. 97, pp. 397–415, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138376211830540X

[5] H. Sultan, A. Chauhan, and S. R. Sarangi, "A survey of chip-level thermal simulators," *ACM Comput. Surv.*, vol. 52, no. 2, apr 2019. [Online]. Available: https://doi.org/10.1145/3309544

[6] 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 Transactions on very large scale integration (VLSI) systems*, vol. 14, no. 5, pp. 501–513, 2006.

[7] M. N. Sabry, "Compact thermal models for electronic systems," *Components and Packaging Technologies, IEEE Transactions on*, vol. 26, pp. 179 – 185, 04 2003.

[8] Z. Yuan, P. Shukla, S. Chetoui, S. Nemtzow, S. Reda, and A. K. Coskun, "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, pp. 1048–1061, 2021.

[9] F. Kaplan, S. Reda, and A. K. Coskun, "Fast thermal modeling of liquid, thermoelectric, and hybrid cooling," in *2017 16th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, 2017, pp. 726–735.

[10] Z. Yuan, G. Vaartstra, P. Shukla, M. Said, S. Reda, E. Wang, and A. K. Coskun, "Two-phase vapor chambers with micropillar evaporators: A new approach to remove heat from future high-performance chips," in *2019 18th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, 2019, pp. 456–464.

[11] S. Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Henkel, and S. Tan, "Post-silicon heat-source identification and machine-learning-based thermal modeling using infrared thermal imaging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, pp. 1–1, 07 2020.

[12] 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, no. 2, pp. 405–419, 2018.

[13] A. Kumar, N. Chang, D. Geb, H. He, S. Pan, J. Wen, S. Asgari, M. Abarham, and C. Ortiz, "Ml-based fast on-chip transient thermal simulation for heterogeneous 2.5d/3d ic designs," in *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2022, pp. 1–8.

[14] J. Lu, J. Zhang, and S. X.-D. Tan, "Real-time thermal map estimation for amd multi-core cpus using transformer," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–7.

[15] C. Knox, Z. Yuan, and A. K. Coskun, "Machine learning and simulation based temperature prediction on high-performance processors," in *International Electronic Packaging Technical Conference and Exhibition*, vol. 86557. American Society of Mechanical Engineers, 2022, p. V001T05A001.

[16] M. Hajikhodaverdian, S. Reda, and A. K. Coskun, "Fast machine learning based prediction for temperature simulation using compact models," in *2025 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2025.

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

[18] L. Chen, W. Jin, and S. X.-D. Tan, "Fast thermal analysis for chiplet design based on graph convolution networks," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 485–492.

[19] R. Ranade, H. He, J. Pathak, N. Chang, A. Kumar, and J. Wen, "A thermal machine learning solver for chip simulation," in *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, ser. MLCAD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 111–117. [Online]. Available: https://doi.org/10.1145/3551901.3556484

[20] J. Wen, S. Pan, N. Chang, W.-T. Chuang, W. Xia, D. Zhu, A. Kumar, E.-C. Yang, K. Srinivasan, and Y.-S. Li, "Dnn-based fast static on-chip thermal solver," in *2020 36th Semiconductor Thermal Measurement, Modeling and Management Symposium (SEMI-THERM)*, 2020, pp. 65–75.

[21] V. A. Chhabria, V. Ahuja, A. Prabhu, N. Patil, P. Jain, and S. S. Sapatnekar, "Thermal and ir drop analysis using convolutional encoder-decoder networks," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 690–696.

[22] D. Bailey, J. Barton, T. Lasinski, and H. Simon, "The nas parallel benchmarks," 08 1993.

[23] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2063384.2063454

[24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: Association for Computing Machinery, 2009, p. 469–480. [Online]. Available: https://doi.org/10.1145/1669112.1669172