Fast Chip Transient Temperature Simulation via Machine Learning

Mohammadamin Hajikhodaverdian*, Sherief Reda†, Ayse K. Coskun*
*Dept. of Electrical & Computer Engineering, Boston University, Boston, MA, USA
{aminhaji, acoskun}@bu.edu
†School of Engineering, Brown University, Providence, RI, USA
sherief_reda@brown.edu

***Abstract*—With growing transistor densities, analyzing temperature in 2D and 3D integrated circuits (ICs) is becoming more complicated and critical. Finite-element solvers give accurate results, but a single transient run can take hours or even days. Compact thermal models (CTMs) shorten the temperature simulation running time using a numerical solver based on the duality between thermal and electric properties. However, CTM solvers often still take hours for small-scale chips because of iterative numerical solvers. Recent work using machine learning (ML) models creates a fast and reliable framework for predicting temperature. However, current ML models demand large input samples and hours of GPU training to reach acceptable accuracy.**

To overcome the challenges stated, we design an ML framework that couples with CTMs to accelerate steady-state and transient thermal analysis without large data inputs. Our framework combines principal-component analysis (PCA) with closed-form linear regression to predict the on-chip temperature directly. The linear regression weights are solved analytically, so training for a grid size of 512×512 finishes in under a minute with only 15–20 CTM samples. Experimental results show that our framework can achieve more than $33\times$ and $49.6\times$ speedup for steady-state and transient simulation of a chip with a 245.95mm^2 footprint, keeping the mean squared error below 0.1°C^2.

***Index Terms*—Thermal simulation, Compact thermal models (CTM), principal-component analysis (PCA), Machine learning, 3D IC**

I. Introduction

With increasing chip transistor density, modern high-performance integrated circuits (ICs) now employ multicore 2D processors and 3D stacked architectures to meet higher computational throughput and memory bandwidth demands. While these designs offer substantial performance benefits, they also introduce significant challenges in thermal management due to increased heat and power densities. These thermal issues can degrade reliability, limit performance, and ultimately reduce the operational lifespan of the chip [1]–[4]. Several techniques, such as dynamic thermal management and temperature-aware design, have been proposed to address thermal challenges. Temperature-aware requires involving the thermal simulation in the design loop, so current thermal simulators are drastically slow, especially for transient, but also even for steady-state. To enable optimizing with realistic power conditions and runtime behavior, we need to substantially reduce thermal simulation time.

979-8-3315-3762-3/25/$31.00 ©2025 IEEE

Conventional thermal simulation methods rely on solving the heat-conduction equations using finite-element methods (FEM) [5]. Commercial tools such as COMSOL and ANSYS use these methods, but they are computationally expensive in terms of runtime and memory consumption. Specifically for transient simulation, FEM-based methods take hours to days to complete a simulation. Compact thermal models (CTMs) were developed to overcome these constraints [6]–[8]. CTMs support steady-state and transient analysis by simplifying thermal modeling and taking advantage of the duality between the thermal and electrical domains. Additionally, they can use advanced cooling methods like heat sinks and microchannels [9], [10]. Iterative numerical solvers are still needed to solve hundreds or thousands of power steps sequentially for transient simulations, which significantly increase computation times, particularly for large-scale 3D designs.

Recent studies have explored machine learning (ML)-based approaches to accelerate temperature prediction for both steady-state and transient simulations [11]–[16]. These models typically use neural networks to learn the relationship between power maps and temperature distributions, often relying on architectures such as convolutional neural networks (CNNs), graph convolutional networks (GCNs), or autoencoders. While they offer promising speedups over traditional solvers, they suffer from several key limitations. Most notably, they require large training datasets, often thousands of samples generated from CTMs or FEM solvers, which can be time-consuming and costly to obtain. Training is compute-intensive and must be repeated from scratch when the floorplan, cooling configuration, or simulation parameters change, with long GPU runs. In some cases, additional data from IR imaging is used [17], [18], further increasing system complexity and limiting practicality.

These challenges highlight the need for a more lightweight and adaptable ML solution. As a result, in this paper, we design a lightweight and data-efficient ML framework (ML-PACT) that integrates seamlessly into existing CTM flows and accelerates both steady-state and transient thermal simulations without requiring large datasets or prolonged training. Our approach applies principal component analysis (PCA) to compress the spatial resolution of power and temperature maps, preserving only the most informative components. We then learn a closed-form linear regression model in the reduced space to predict future temperature states. This approach

requires only 15–20 CTM samples and completes training in under a minute. Once trained, the model can efficiently predict steady-state temperatures or simulate long transient sequences with minimal overhead.

Our contributions are summarized as follows:

- We build a unified ML-based framework that supports both steady-state and transient thermal simulation for 2D and 3D chip architectures, eliminating the need for separate model setups.
- We significantly reduce conventional ML models' data and training requirements by using a Koopman-inspired linear formulation, requiring only 15–20 training samples to achieve high accuracy.
- We introduce a PCA-based dimensionality reduction method that enhances scalability across varying grid sizes, enabling efficient and accurate thermal predictions even for high-resolution simulations.

Compared to a conventional CTM baseline, our framework achieves consistent speedups of over $33\times$ and $43\times$ for steady-state and transient simulations across both 2D and 3D chips with a $245.95\mathrm{mm}^2$ footprint. These results demonstrate the potential of our method to enhance traditional thermal modeling flows with fast, accurate, and scalable prediction capabilities.

The remainder of the paper is organized as follows: Section II reviews related work. Section III describes our lightweight approach. Section IV presents experimental results and comparisons. Section V concludes the paper.

II. Related Work

ML methods appear as a fast alternative to CTMs and FEM-based solvers for thermal simulation. These ML-based methods offer faster inference while maintaining high accuracy, making them suitable for accelerating thermal analysis during design and runtime. Some studies use infrared (IR) camera data from fabricated chips to train their models [17]–[19]. Although this approach achieves excellent accuracy by learning directly from measured silicon, it relies on costly IR imaging infrastructure and post-silicon characterization, making it unsuitable for early design-stage prediction and limiting broader use. To overcome this, recent works focus on simulation-based ML models that use synthetic training data generated from CTM or FEM solvers. These ML models utilize various architectures such as GCN and encoder-decoder CNNs [13]–[15], [20], [21], which are suitable for the pre-silicon design phase. While they provide accurate and fast thermal prediction, their major limitation is their need for large training datasets and long GPU training runs. These datasets are usually generated through thousands of CTM or FEM runs, which are both time-consuming and computationally expensive. Moreover, these models often require retraining when there are changes in floorplans, grid sizes, or thermal boundary conditions, making them less flexible. For example, Ranade et al. [20] trained separate models for different chip configurations, each requiring thousands of CTM samples. Wen et al. [21] predicted temperature differences instead of absolute values and had to rely on additional FEM simulations at inference time to reconstruct the complete temperature field. Similarly, FaStTherm [13] proposed a deep autoencoder framework that models thermal dynamics in a latent space, enabling long-term rollouts and improved inference stability. However, FaStTherm still demands many training samples, long GPU training times, and full retraining for each new scenario, resulting in limited scalability and slow turnaround time in iterative design processes.

Compared to prior methods that rely on large datasets and long training times, our framework offers a fast and lightweight alternative by combining PCA-based dimensionality reduction with closed-form linear regression. This approach significantly reduces model complexity, enabling training in under a minute using only a small number of CTM samples. As a result, the model can be quickly retrained when power profiles or design configurations change. Moreover, it supports both steady-state and transient analysis within a single unified model, making it practical and adaptable across different chip designs and thermal scenarios.

III. Temperature Prediction Using PCA-based Machine Learning and CTMs

In this paper, we design a lightweight machine learning framework to accelerate compact thermal simulations in CTMs using PCA-based dimensionality reduction and closed-form linear regression. Our method addresses conventional solvers' runtime and scalability limitations by introducing a fast, data-efficient ML model that requires minimal setup and training time. First, we describe the formulation of temperature simulation used in CTMs. Then, we outline our PCA-based learning framework, including how we structure the input and output data. Finally, we detail the approach to select the number of PCA components to balance accuracy and efficiency.

A. Temperature Simulation Formulation

The CTM constructs an equivalent thermal circuit by leveraging the duality between heat flow and electric current [6], [8]. For steady-state, this network reduces to a linear system:

$$\mathbf{G}\,\mathbf{T} \;=\; \mathbf{P}, \tag{1}$$

where $\mathbf{T}$ is the vector of node temperatures (K), $\mathbf{G}$ is the thermal conductance matrix (W/K), and $\mathbf{P}$ is the vector of power dissipation (W). In realistic high-resolution designs, $\mathbf{G}$ becomes extremely large and sparse, making even this seemingly simple linear solve computationally demanding. Iterative methods are typically employed, but the cost grows rapidly with the number of thermal nodes, often becoming the bottleneck in the thermal analysis pipeline.

For transient simulations, CTMs include the lumped thermal capacitance at every node to capture temporal behavior. Let $\mathbf{C}$ be the diagonal heat-capacity matrix (J/K). The model is:

$$\mathbf{C}\,\frac{d\mathbf{T}(t)}{dt} \;+\; \mathbf{G}\,\mathbf{T}(t) \;=\; \mathbf{P}(t), \tag{2}$$

where $\mathbf{C}$ is a diagonal matrix representing nodal heat capacities (J/K), and $\mathbf{T}(t)$, $\mathbf{P}(t)$ denote the time-varying temperature

and power vectors, respectively. This formulation requires solving an extensive linear system at every time step, and for fine-grained temporal resolution, the cumulative cost becomes impractical. These limitations motivate the need for alternative formulations that retain physical accuracy while reducing computational overhead, particularly in high-resolution 3D simulations, which take hours of simulation even in CTMs.

B. Framework Design

Instead of solving (1) or (2) at every step, we map the power–temperature inputs into a low-dimensional latent space where also the transient simulation becomes linear based on the Koopman theorem [22], which states a nonlinear system can be switched to a linear system with a set of transformation. Unlike prior work such as FaStTherm [13], which uses a deep autoencoder to learn this transformation, we adopt PCA, offering a far simpler and computationally efficient alternative. Although PCA does not compute the full Koopman operator and our model is Koopman-inspired linear latent-space approximation, prior work by Ranieri et al. [23] showed that the dominant principal components capture more than 95% of the spatial variance in temperature distributions. This insight makes PCA particularly well-suited for high-resolution thermal modeling with minimal loss in accuracy. Working in this latent space also avoids a significant cost in steady-state runs. A direct approach for steady-state simulation would invert the conductance matrix $\mathbf{G}$. However, the size of $\mathbf{G}$ grows quadratically with spatial resolution, and its explicit inversion quickly becomes impractical. By working in the reduced PCA space, our framework avoids this bottleneck by solving a smaller linear problem, achieving the same result with a fraction of the computational cost.

To formulate the model in latent form, let $\hat{\mathbf{T}}_k$ and $\hat{\mathbf{P}}_k$ denote the PCA-coefficient vectors of the temperature and power fields at time step k. The one-step transient update is

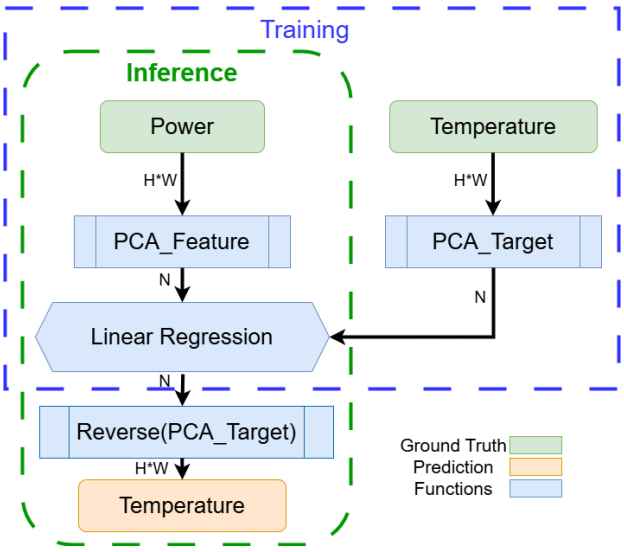
$$\hat{\mathbf{T}}_{k+1} = \mathbf{W} \begin{bmatrix} \hat{\mathbf{P}}_{k+1} \\ \hat{\mathbf{T}}_k \end{bmatrix} + \mathbf{b}, \tag{3}$$

where the weight matrix $\mathbf{W}$ and bias $\mathbf{b}$ are estimated once from a small training set. For steady-state conditions, the relation reduces to

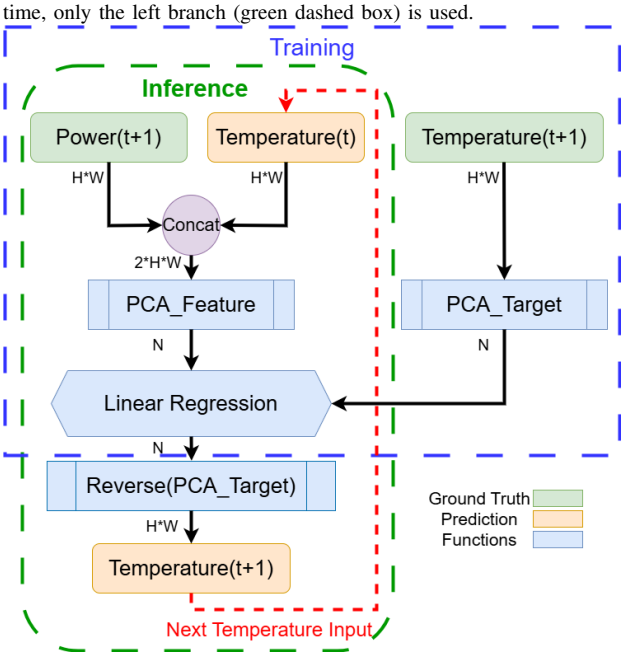
$$\hat{\mathbf{T}} = \mathbf{W}_s \hat{\mathbf{P}} + \mathbf{b}_s. \tag{4}$$

Eq. (3) and (4) allow us to advance the system in time or obtain static temperatures directly, eliminating the expensive iterative solvers that dominate conventional CTM workflows.

Fig. 1 depicts the entire PCA-driven framework in both steady-state (top) and transient (bottom). During training, the CTM reference temperatures shown in green supervise the model to calculate Eq. (3) and (4) weights and biases. We collect a small set of samples (10 to 15) from CTM, train our feature and target PCAs using them. We use the closed-form in the PCA latent space to get the weight and bias we need for the linear regression (Dashed blue outlines mark elements that exist only while training, whereas dashed green outlines track



(a) Steady-state case. At training time, the model pairs each power map with its CTM temperature to fit the regression weights; at inference time, only the left branch (green dashed box) is used.



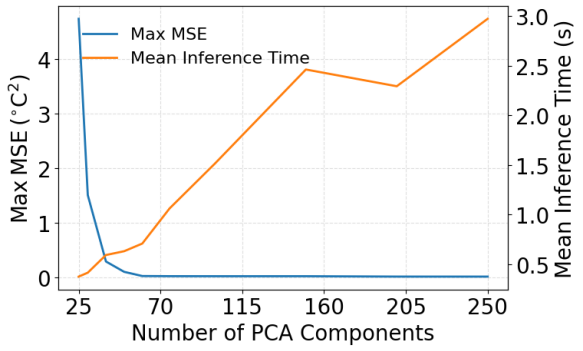
(b) Transient case. For training, the model learns a regression by using CTM temperatures. For inference, the model receives the next-step power map and the current temperature, and predicts the temperature at t+1. The predicted temperature is then fed back as the next input, enabling fast roll-out over long traces without additional CTM calls.

Fig. 1: Overview of the designed PCA-based framework: (a) steady-state workflow, (b) transient roll-out workflow.

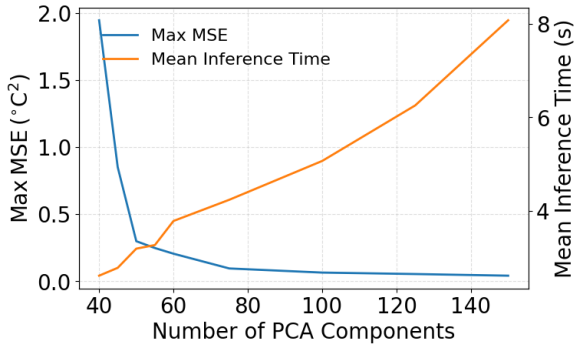
the inference-time path). For inference, the reference inputs from CTM will be disconnected, and the framework outputs its own temperature estimates, highlighted in orange. The flow traverses three blue-bordered modules (PCA embedding, a closed-form linear regressor, and inverse PCA reconstruction), which provide a deterministic round-trip between the original $H \times W$ grid and a compact latent space N, which is the number of PCA components. In the transient path, the temperature predicted at time step k is concatenated with the power map

at $k+1$ and routed back through the same fixed modules, enabling fast multi-step roll-out with no additional CTM solves. In this case, while predictions are recursive, error does not accumulate significantly. Temperature is mostly driven by current power, with limited dependence on previous steps due to localized time constants. Neighboring cells also help smooth out variations, keeping predictions stable.

While prior works have used deep networks like CNNs or autoencoders for thermal modeling, such complexity is unnecessary in our case. The spatial distribution of on-chip temperature is inherently smooth due to the diffusive nature of heat conduction, where high-frequency variations are naturally dampened as heat spreads over space. This diffusion smoothness is especially pronounced under the uniform material properties assumed in our setup, allowing a small number of spatial modes to capture most of the variance. PCA is well-suited to leverage this structure, capturing the dominant temperature patterns with only a few components. Similar to most ML methods, our framework does not generalize across arbitrary floorplan or boundary condition changes, and retraining is needed when the physical setup is altered. However, due to the lightweight nature of our model and minimal data needs, this retraining remains fast and inexpensive, making frequent updates practical without compromising efficiency or scalability.



(a) Max MSE and Mean Inference Time vs PCA Components: 2D.



(b) Max MSE and Mean Inference Time vs PCA Components: 3D.

Fig. 2: Trade-off among PCA latent dimension, maximum MSE (left axis, blue), and mean inference time per sample (right axis, orange) for (a) 2D and (b) 3D floorplans. A modest latent size ($\sim$50 components for 2D, $\sim$60 components for 3D) keeps the error near zero while minimizing runtime.

Simulation Type	Steady-State	Transient
Solver	KLU	Trap
Number of Timesteps	1	284, 500
Grid Size	100×100, 512×512	100×100, 128×128
Ambient Temperature	45°C	45°C
CPU Cores Used	16	16
Active Layer Thickness (2D)	0.10 mm	0.10 mm
Active Layer Thickness (3D)	91.44 μm	91.44 μm

TABLE I: PACT configuration simulation parameters.

C. PCA Component Selection

As discussed in Section III-B, the dimensionality reduction introduced by PCA introduces a single key hyperparameter: the number of principal components, denoted as N. This value determines the size of the latent space and directly controls the trade-off between accuracy and computational efficiency. To select an appropriate value for N, we conduct a sweep across different component counts and evaluate both the maximum mean squared error (MSE) and the average inference time. Fig. 2 presents the results for both the 2D scenario (2a) and the 3D scenario (2b). In both cases, we observe a steep decrease in maximum MSE as the number of components increases from 10 to approximately 50–70. Beyond this range, the gains in accuracy diminish, and the curve begins to saturate. At the same time, the inference time grows roughly linearly with N, reflecting the increasing size of the latent-space computations. Finding a balance between accuracy and throughput is critical for practical inference. Selecting too few components degrades reconstruction quality, while selecting too many reduces the performance benefit of the PCA-based framework. Based on this trade-off, we adopt $N = 50$ for both the 2D and 3D settings. This choice lies at the "elbow" of the MSE curve, where additional components yield minimal accuracy improvement. At this setting, our model keeps the worst-case temperature error under $0.5°C^2$ while maintaining a few seconds of inference runtime, making it well-suited for fast, high-resolution thermal simulations.

IV. Evaluation

In this section, we first explain our experimental setup for the framework, followed by a detailed set of results to demonstrate our method's benefits.

A. Experimental Setup

We evaluate our framework on both 2D and 3D chip architectures. For the 2D case, we use the block-level floorplan of an Intel Core i7-6950X processor. Power traces are generated by running ten NAS parallel benchmark applications [24] using Sniper [25] and McPAT [26]. We create the power traces required for thermal simulation in CTM by instantiating each component (e.g., core0, core1, etc.) in the McPAT input configuration, enabling detailed power estimation across all cores and system components. For the 3D case, we consider a monolithic design with three active layers, including one compute layer and two memory layers. In both cases, the total chip footprint is fixed at 16.8 mm × 14.64 mm, and

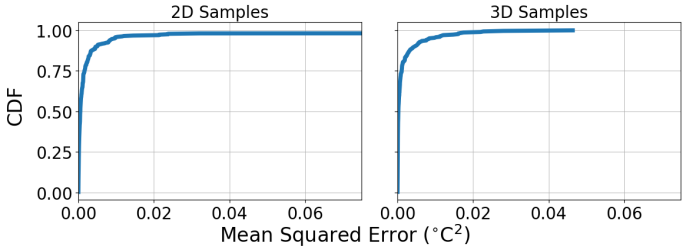


Fig. 3: CDF of MSE for steady-state prediction on (a) 2D and (b) 3D test cases (512×512 grid, 375 test samples each).

the peak power used during training reaches up to $140\,\mathrm{W}$. To ensure our model can generalize to realistic high-power scenarios, we increase the input power during inference to trigger thermal hotspots above $130°\mathrm{C}$. Our training framework relies on CTMs, and we use PACT [8] to generate training labels. We are comparing our results with PACT, a recent CTM-based tool that has been shown to be faster than other state-of-the-art simulators. The details of the configuration parameters of PACT for our test cases are summarized in Table I. We evaluate 2D and 3D architectures across various simulation resolutions, from 100×100 to 512×512 grids, to demonstrate our model's scalability to problem size. The monolithic 3D testcase consists of three active layers, each separated by dielectric layers and metal routing (BEOL), and is mounted on a bulk silicon substrate. The total stack height reaches 91.44μm, capturing the heat propagation challenges of monolithic 3D integration.

In our experiments, we report simulation time as a metric for computational efficiency. The simulation time of our framework consists of training time, simulation time of PACT for training sets, and inference time. We use the mean squared error (MSE) of all the chip temperature predictions to demonstrate our model's accuracy compared to the PACT ground truth. For transient test cases, we are reporting the MSE of the timestep with the maximum value over all timesteps to show the worst-case prediction of our model in a transient simulation. For a fair comparison of simulation time, we utilize 16 CPU cores without utilizing any GPU cores for both PACT and our framework. Finally, to demonstrate our model's fast training setup, we select FaStTherm [13] to compare their model's training time and accuracy with ours. In this experiment, we utilize 2272 training samples (8 transient simulations with 284 timesteps) and train the FaStTherm model for 2000 epochs using one NVIDIA L40S.

B. Results

We start by evaluating the performance of our framework on steady-state simulations. Our dataset consists of 390 samples for each 2D and 3D scenario, using a grid size of 512×512. From each scenario, we use 15 samples to train the model. Training samples are selected empirically by increasing the size of the training set to reduce the error. The cumulative distribution function (CDF) of the MSE on the remaining test samples is shown in Fig. 3. The results indicate that our model maintains high accuracy, with MSE values below $0.08°\mathrm{C}^2$ for

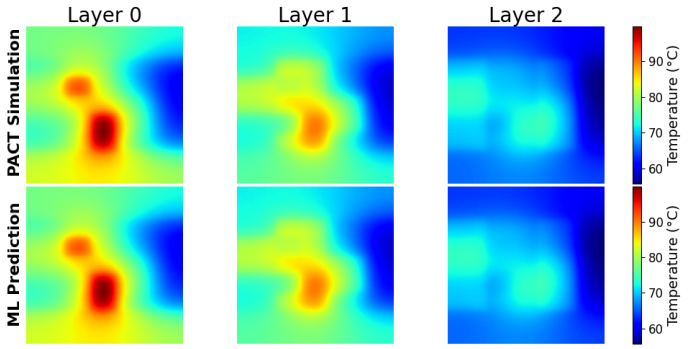


Fig. 4: PACT and our ML prediction for each layer of an input sample of monolithic 3D design.

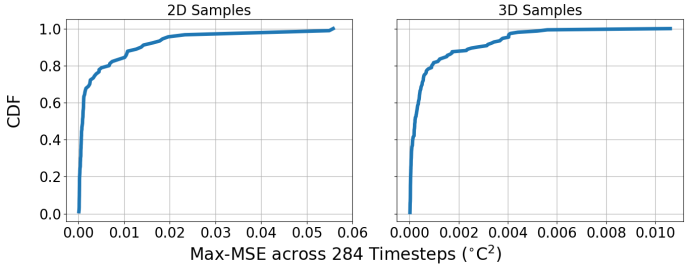


Fig. 5: CDF of MSE for transient prediction on (a) 2D with 10 training samples and (b) 3D with 8 training samples (100×100 grid).

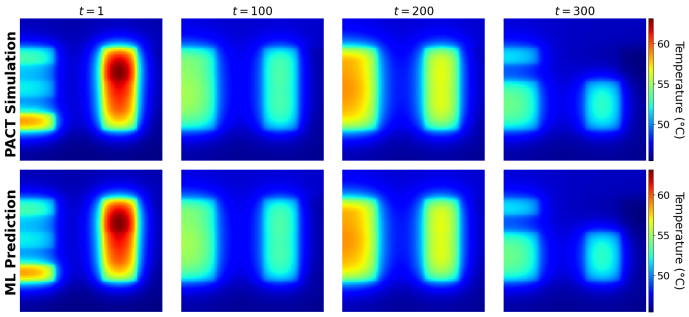


Fig. 6: PACT and our ML prediction for different time steps of an input sample for transient simulation.

2D and $0.05°\mathrm{C}^2$ for 3D. Fig. 4 further illustrates this accuracy by comparing the temperature predictions of our model against those of the CTM-based PACT framework for a representative 3D test sample. The close match between the two outputs confirms the effectiveness of our approach in replicating CTM-level accuracy.

For transient simulations, we evaluate 390 samples for each 2D and 3D scenario, using a grid size of 100×100 with 284 timesteps per sample. To train our framework, we simulate 10 samples for 2D and 8 for 3D using the CTM-based PACT solver, resulting in 2840 and 2272 temperature maps, respectively. These samples are used to learn the transient dynamics of temperature changes. Despite the small number of training samples, our model achieves efficient convergence, with training times of only 7.07 seconds for the 2D case and 18.11 seconds for the 3D case. Fig. 5 presents the CDF of the maximum MSE observed across all timesteps within each

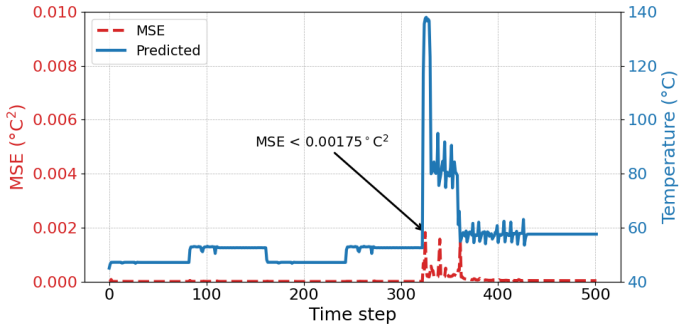


Fig. 7: Temperature prediction vs. ground truth at a specific point with per-timestep MSE shown as bars.

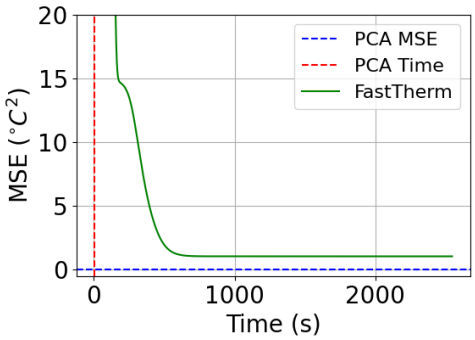


Fig. 8: MSE vs. Training Time for our framework (PCA-based) and FaStTherm Methods.

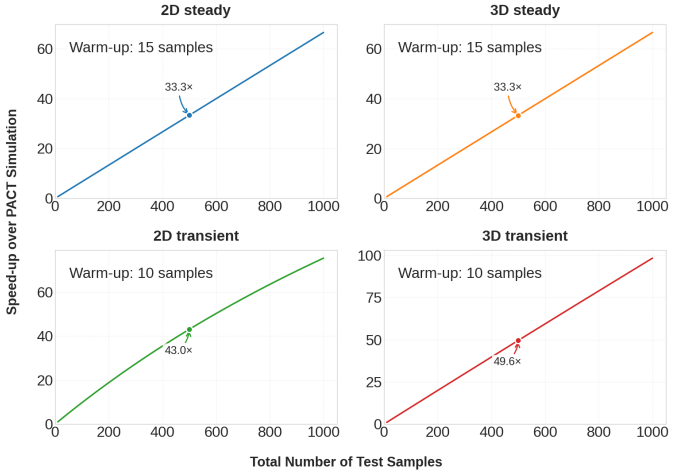


Fig. 9: Speedup of our framework over the PACT simulation baseline as a function of the total number of test samples. Our method achieves up to $49.4\times$ acceleration for transient 3D cases after a short warm-up.

test sample. The results indicate strong predictive performance, with the maximum MSE remaining below $0.06°\mathrm{C}^2$ for the 2D scenario and $0.015°\mathrm{C}^2$ for the 3D scenario. To further demonstrate accuracy, Fig. 6 compares temperature predictions from our model with the corresponding outputs from PACT across selected timesteps. The close visual and numerical match demonstrates that our model effectively captures the temporal temperature behavior. To evaluate the robustness of our model under dynamic workloads, we further simulate a power input with frequent and abrupt changes. As shown in Fig. 7, our model effectively captures rapid thermal responses, achieving a worst-case MSE as low as $0.00175°\mathrm{C}^2$ despite the increased temporal complexity and increasing the temperature to up to $140°\mathrm{C}$. These results highlight our framework's accuracy and generalization across several transient scenarios.

To evaluate the efficiency of our framework against FaStTherm, we compare the changes in MSE over training time. As shown in Fig. 8, FaStTherm takes over 2000 seconds to reach an MSE close to that of our PCA-based model. In contrast, our framework achieves a significantly lower final MSE in just under 2 seconds of training, as indicated by the horizontal and vertical dashed blue and red lines. This substantial speedup results from using a closed-form linear model in the PCA space, eliminating the need for iterative gradient-based optimization. More importantly, the accuracy of our approach is not compromised by this speed; the resulting MSE remains well within an acceptable range. Combined with the strong results from steady-state and transient evaluations, this comparison further highlights the practicality of our framework in real-world design scenarios where fast and reliable thermal estimation is essential.

We assess the efficiency of our method by comparing the full end-to-end runtime with the CTM-based PACT simulator, including both the training (warm-up) and inference phases. Fig. 9 illustrates the achieved speedup as a function of the total number of test samples following a brief training period. Across all evaluated scenarios, our framework consistently demonstrates substantial improvements in runtime performance. For steady-state simulations, it delivers a $33.3\times$ speedup in both 2D and 3D cases with only 15 warm-up samples, showing strong performance even with limited data. The gains are even more pronounced in transient simulations, which are typically more computationally intensive: our method achieves $43\times$ and $49\times$ speedups for the 2D and 3D settings, respectively, using just 10 warm-up samples. These results demonstrate that our method can significantly accelerate simulation workloads once the initial training is completed, especially in large-scale or iterative design flows.

V. Summary

This work presented a fast and accurate ML framework for steady-state and transient temperature prediction in 2D and 3D ICs called ML-PACT. By leveraging CTMs and dimensionality reduction through PCA, our approach enables low-cost inference while preserving high spatial resolution. Unlike other ML solutions requiring extensive training data, our framework achieves competitive accuracy with only a few samples. Experimental results demonstrate that our model outperforms prior approaches in inference speed, achieving up to $49\times$ speedup over numerical solvers, while maintaining low prediction error across various workloads. These results highlight the potential of our method to serve as a lightweight, plug-in substitute for thermal estimation in chip design flows.

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] L. Zhu and S. K. Lim, "Invited: Design automation needs for monolithic 3d ics: Accomplishments and gaps," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–4.

[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] K. Dhananjay, P. Shukla, V. F. Pavlidis, A. Coskun, and E. Salman, "Monolithic 3d integrated circuits: Recent trends and future prospects," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 837–843, 2021.

[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] 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 (DATE)*, 2025, pp. 1–2.

[12] 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.

[13] T. Zhu, Q. Wang, Y. Lin, R. Wang, and R. Huang, "Fasttherm: Fast and stable full-chip transient thermal predictor considering nonlinear effects," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[14] 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.

[15] 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 *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 690–696. [Online]. Available: https://doi.org/10.1145/3394885.3431583

[16] M. Hajikhodaverdian, S. Reda, and A. K. Coskun, "Steady-state temperature prediction based on compact thermal models using machine learning," in *2025 24th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, 2025.

[17] 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.

[18] W. Jin, S. Sadiqbatcha, J. Zhang, and S. X.-D. Tan, "Full-chip thermal map estimation for commercial multi-core cpus with generative adversarial learning," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[19] 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.

[20] 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

[21] 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.

[22] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, pp. 1307–1346, 2015.

[23] J. Ranieri, A. Vincenzi, A. Chebira, D. Atienza, and M. Vetterli, "Eigenmaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors," in *DAC Design Automation Conference 2012*, 2012, pp. 636–641.

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

[25] 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

[26] 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

VI. Artifact Appendix

A. Abstract

This artifact provides the Python implementation for the ML-PACT framework, a lightweight machine learning model designed to accelerate steady-state and transient thermal simulations for 2D and 3D integrated circuits of our previous compact thermal simulator (CTM), PACT. The artifact provides scripts for training the principal component analysis (PCA) based linear regression model and for executing inference to predict chip temperature maps. The workflow also includes scripts to generate the necessary training and test samples for the model by running the PACT simulator with different input power traces. The complete artifact is available at: https://github.com/peaclab/PACT/tree/master/MLPACT.

B. Artifact check-list (meta-information)

- **Algorithm:** "Principal Component Analysis (PCA)", "Closed-Form Linear Regression"
- **Program:** "Python script"
- **Transformations:** "PCA (scikit-learn), linear projection"
- **Model:** "Trained PCA components and linear regression weights"
- **Hardware:** "Any multi-core CPU (experiments were conducted on a 16-core system)"
- **Metrics:** "MSE ($^{\circ}C^2$), rollout inference time (s)"
- **Output:** "Per-test predicted temperature maps as CSV (and optional PNG heatmaps)"
- **Proprietary EDA tools:** "PACT and Xyse for creating the train set, which are both open-source"
- **How much disk space required (approximately)?:** "Depends on the number of samples (20GB to 200 GB)"
- **Publicly available?:** "Yes"
- **Code licenses (if publicly available)?:** "GNU GPLv3"

C. Description

- **How to access:** The artifact is available for download from the public GitHub repository: https://github.com/peaclab/PACT/tree/master/MLPACT.
- **Hardware dependencies:** A standard multi-core CPU is required. The experiments in the paper were conducted on a 16-core system.
- **Software dependencies:** The framework requires Python 3.8+ and the packages listed in the `requirements.txt` file, such as `pandas`, `torch`, and `scikit-learn`. The PACT simulator is also required to generate new datasets.
- **Commercial software dependencies:** None.
- **Data sets:** The datasets consist of power trace files, which transform to the proper input format (`.cir`) and the corresponding ground-truth temperature maps (`.csv`). The data generation workflow is illustrated in Fig. 10. User inputs are processed by PACT to generate a small training set (10-15 samples) of input/output pairs, while the rest of the inputs are prepared for the test set.

D. Installation

1) Clone the artifact repository from the public GitHub URL.
2) Install the required Python packages using the provided `requirements.txt` file. It is recommended to use a virtual environment.

```
pip install -r requirements.txt
```

3) (Optional) To generate new training or test datasets, the PACT simulator must be installed. Please follow the installation instructions provided in the PACT source repository.

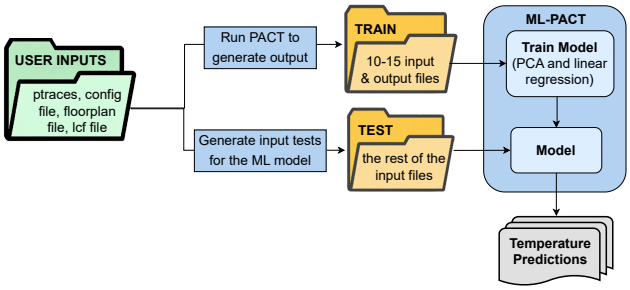


Fig. 10: Data generation and ML-PACT workflow.

E. Experiment workflow

The provided script does the following:

1) Parse `Train/*.cir` to power tensors and `Train/*.cir.csv` to temperature tensors.
2) Build samples: feature $= [P_{t+1}, T_t]$, target $= T_{t+1}$.
3) Standardize, apply PCA (features/targets), solve linear regression.
4) For each `Test/*.cir`, roll out predictions for simulation steps from a uniform initial map (default $318.15\,\mathrm{K}$).
5) Save per-file CSV to output directory (and optional PNGs).

F. Evaluation and expected results

Running the included sample results:

- Console prints of training MSE in the target PCA space and in the original (scaled) space.
- Per-test rollout inference time.
- CSV outputs.

Exact numbers vary with grid size, PCA dimensions, and number of training pairs; training and inference are complete in minutes on a CPU if PACT output is already generated.

G. Experiment customization

You can refer to the `ReadMe.md` file available in the GitHub Repository.

H. Methodology

Submission, reviewing, and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-and-badging-current
- http://cTuning.org/ae/submission-20201122.html
- https://github.com/ml-eda/artifact-evaluation/