Recursive weight-sharing transformers for semiconductor thermo-mechanical reliability

Kart-leong Lim
Institute of Microelectronics (IME),
*Agency for Science, Technology and Research (A*STAR),* Singapore
limkl@a-star.edu.sg

arXiv:2607.27251v1 [cs.LG] 28 Jul 2026

***Abstract*—Transformer-based surrogate models are increasingly used to replace expensive first-principles simulation in engineering design. But conventional transformer architectures are often over parameterized for the small, low-dimensional datasets typical of engineering design spaces, where large simulation data is expensive to generate. Under these conditions, excess parameter capacity leads to overfitting rather than improved accuracy, while also incurring unnecessary memory and compute overhead — this motivates a shift towards architectures that focus on additional compute rather than additional learnable parameters. This paper presents a hardware-aware evaluation of three recursive transformer paradigms for surrogate thermo-mechanical analysis of advanced packages: a)Tiny Recursive Model, b) our proposed Depth Recursive transformer, c) and a simple recursive transformer. We systematically compare their predictive performance (Recall@K, Mean Reciprocal Rank), parameter count, computational complexity (FLOPs), providing practical design guidelines for selecting recursive transformer architectures under resource-constrained scenarios. We validate this principle on two low-dimensional engineering prediction tasks: 1) thermo-mechanical reliability analysis of advanced semiconductor packages, where stress and warpage from thermal cycling must be evaluated repeatedly across a design-of-experiments sweep under costly finite element analysis (FEA). 2) Laplace PDE iterative numerical solver for capacitance field. Overall, recursive weight-sharing transformers provide an effective and generalizable trade-off between prediction accuracy, parameter efficiency, and computational cost for small-data engineering surrogate modeling — a regime for which conventional large-parameter transformers are poorly suited — as demonstrated on advanced package reliability prediction and capactior electrostatic field modeling.**

I. Introduction

Transformer models have demonstrated strong predictive performance across a wide range of engineering applications, including electronic design automation (EDA) [1], scientific machine learning [2], and digital twins [3]. However, their increasing computational and memory requirements present significant challenges for deployment in hardware-constrained environments, such as on-chip inference, edge accelerators, and real-time design optimization [4], [5]. Consequently, substantial research has focused on reducing transformer computation through improvements at the algorithmic, architectural, and system levels [6], [7], [8], [9]. Thermo-mechanical simulation remains computationally expensive, motivating the use of transformer-based surrogate models for rapid reliability prediction. However, deploying such models within practical EDA workflows requires architectures with low computational complexity, compact memory footprints, and high inference throughput. These requirements have motivated a broad range of transformer compute reduction techniques and can be broadly categorized into seven complementary paradigms according to the aspect of computation they optimize. These include **Efficient Attention** [10], [6], **Token Reduction** [11], [12], **Conditional Computation** [7], [13], **Architectural Optimization** [8], [14], [15], [16], **Model Compression** [9], [17], [18], **Hardware Optimization** [19], [20], and **State Space Models** [21], [22]. Although these approaches pursue the common objective of improving computational efficiency, they address different computational bottlenecks, ranging from reducing attention complexity and sequence length to redesigning network architectures or replacing attention-based models altogether. Among these paradigms, this work focuses on **Architectural Optimization**, where computational efficiency is achieved by redesigning the transformer itself. In particular, recursive weight sharing enables deeper computation without a proportional increase in parameters [8], [15], [16], making it particularly attractive for resource-constrained EDA applications. Although numerous approaches have been proposed to reduce transformer computation, their suitability depends on the target application. Efficient attention mechanisms primarily benefit long-sequence models where self-attention dominates the computational cost [10], [6], while token reduction techniques rely on large token sets to achieve meaningful computational savings [11], [12]. Conditional computation and Mixture-of-Experts architectures improve scalability for large foundation models but introduce additional routing complexity and memory overhead [7], [13], [23]. Model compression and hardware-specific optimizations are generally applied after model design or target specific execution platforms [9], [18], [19]. In contrast, our application involves compact transformer surrogates operating on short input sequences under strict memory and computational constraints. Consequently, Architectural Optimization provides the most suitable design strategy, enabling computation to be reduced intrinsically through recursive weight sharing while preserving a compact, hardware-friendly architecture [8], [15], [16].

II. Recursive weight sharing transformers

The main concept of recursive transformer is to treat transformer as a reusable template (e.g. RECUR) throughout the model. Multiple replicas of RECUR are used in a model, but

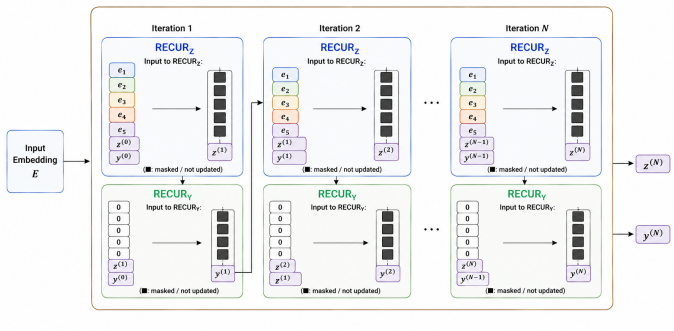

Fig. 1: Tiny Recursive model

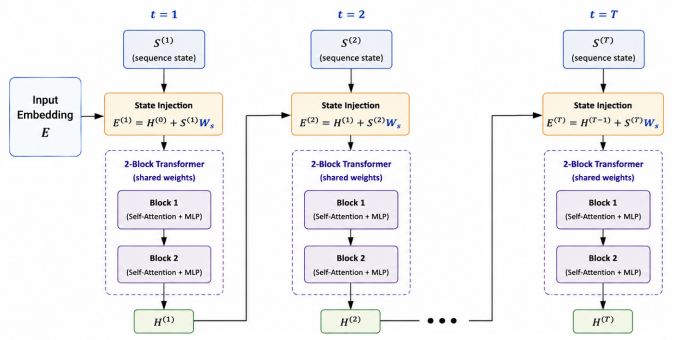

Fig. 2: Depth Recursive model

they all share the single set of weights, which is updated once during each training iteration. RECUR should be independent of models we choose to deploy in. The simplest configuration of RECUR is a one block transformer in Eqn.1 which consists of Self-Attention ATTN$(\cdot)$, Normalization layer NORM$(\cdot)$, Feed-Forward Network (FFN). Conversely, if we want a deeper transformer, we can define RECUR using e.g. two transformer blocks.

$$\begin{aligned}\text{RECUR} = \ & \text{ATTN}(\cdot \mid W_Q, W_K, W_V, W_O) \\ & \to \text{ADD}_1 + \text{NORM}_1(\cdot \mid \gamma_1, \beta_1) \\ & \to \text{FFN}(\cdot \mid W_1, W_2) \\ & \to \text{ADD}_2 + \text{NORM}_2(\cdot \mid \gamma_2, \beta_2).\end{aligned} \tag{1}$$

A. Tiny Recursive Model

The Tiny Recursive Model (TRM) uses weight sharing RECUR to cap the parameters while perform AI reasoning on Sudoku and puzzle tasks shown in Fig. 1. Within each iteration, RECUR_Z updates the latent feature $\mathbf{z}$ by feeding the input embeddings $\mathbf{E} \in \mathbb{R}^{L \times d_{token}} = [\mathbf{e}_1, \ldots, \mathbf{e}_L]$–where L is number of tokens and d_{token} is the embedding dimension–along with the previous latent feature $\mathbf{z}^{(n-1)}$, and the previous output feature $\mathbf{y}^{(n-1)}$. The updated latent feature $\mathbf{z}^{(n)}$ is then passed to RECUR_Y, which updates the output representation $\mathbf{y}^{(n)}$ while masking the input embeddings with zeros. Repeating this coupled update for N times defines the LATENT operation in Eqn.2. This progressively refine both $\mathbf{z}^{(n)}$ and $\mathbf{y}^{(n)}$.

$$\begin{aligned}\text{LATENT} &\triangleq (\text{RECUR}_Y \circ \text{RECUR}_Z)^N \\ \mathbf{z}^{(n)} &= \text{RECUR}_Z\left([\mathbf{e}_1, \ldots, \mathbf{e}_L, \mathbf{z}^{(n-1)}, \mathbf{y}^{(n-1)}]\right), \\ \mathbf{y}^{(n)} &= \text{RECUR}_Y\left([\mathbf{0}, \ldots, \mathbf{0}, \mathbf{z}^{(n)}, \mathbf{y}^{(n-1)}]\right).\end{aligned} \tag{2}$$

Finally to compute TRM (Eqn.3), LATENT is executed for T times and fed into another RECUR. Increasing either the inner N or the outer T loops increases the computation cost (FLOPs) without increasing the parameters through weight sharing.

$$\begin{aligned}&\text{TRM}\left(\mathbf{E}, \mathbf{z}^{ini}, \mathbf{y}^{ini}\right) \triangleq \text{RECUR} \circ (\text{LATENT})^T \\ &= \text{RECUR}\left(\mathbf{E}, \left[\left(\text{RECUR}_Y\left(\mathbf{0}, \text{RECUR}_Z\left(\mathbf{E}, \mathbf{z}, \mathbf{y}\right), \mathbf{y}\right)\right)^N\right]^T\right).\end{aligned} \tag{3}$$

B. Depth Recursive Model

From our domain expertise in advanced packaging, we observe that the dataset exhibits a gradually changing depth-like structure for each fixed design case. Motivated by this observation, we repurpose RECUR by treating sequential depth as a state input while fully retaining the weight sharing objective in Fig.2 We call this approach as the Depth Recursive Model (DEPTH). Inspired by Recurrent Neural Networks (RNNs), DEPTH adapts the notion of hidden states to weight-dependent sequential depth states within RECUR. Using the terminology of RNN for sequential modeling, in DEPTH (Eqn.4) we define sequential state as $S^{(t)}$, initial hidden state as embedding input $\mathbf{E} = \mathbf{H}^{(0)}$, and sequential (or recursive) output $H^{(t+1)}$ for depth $t \in \{1, \ldots, T\}$.

$$\begin{aligned}&\text{DEPTH}\left(\mathbf{E}_{S^t \nsubseteq E}, \mathbf{S}^{(1)}, \ldots, \mathbf{S}^{(T)}\right) \triangleq \left(\text{RECUR} \circ \left(\cdot + \mathbf{S}^{(t)}\mathbf{W}_s\right)\right)^T \\ &= \text{RECUR}\left(\mathbf{E} + \mathbf{S}^{(1)}\mathbf{W}_s\right) \to \mathbf{H}^{(1)} \\ &\to \text{RECUR}\left(\mathbf{H}^{(1)} + \mathbf{S}^{(2)}\mathbf{W}_s\right) \to \mathbf{H}^{(2)} \\ &\cdots \to \text{RECUR}\left(\mathbf{H}^{(T-1)} + \mathbf{S}^{(T)}\mathbf{W}_s\right) \to \mathbf{H}^{(T)}.\end{aligned} \tag{4}$$

C. Simple Recursive model

The simple recursive model (SIMPLE) in Fig.3 is seen to repeatedly apply an identical recursive transformation. Using (Eqn. 5), SIMPLE applies RECUR for T recursive iterations to generate the output $H^{(T)}$.

$$\begin{aligned}\text{SIMPLE}(\mathbf{E}) &\triangleq (\text{RECUR})^T \\ &= \text{RECUR}(\mathbf{E}) \to \mathbf{H}^{(1)} \\ &\to \text{RECUR}\left(\mathbf{H}^{(1)}\right) \to \mathbf{H}^{(2)} \\ &\cdots \to \text{RECUR}\left(\mathbf{H}^{(T-1)}\right) \to \mathbf{H}^{(T)}\end{aligned} \tag{5}$$

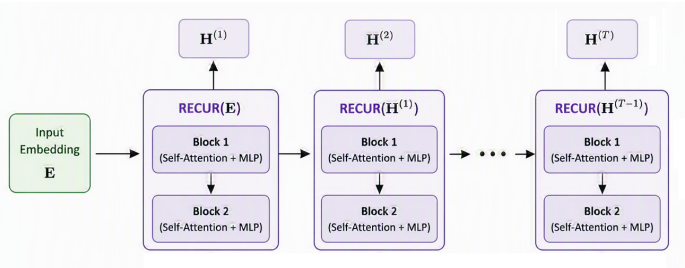

Fig. 3: Simple Recursive model

III. Experimental Results

A. Baselines and proposed models

To thoroughly evaluate the tracking and retrieval performance, we compare our approach against seven baseline configurations categorized across four distinct architectural paradigms:

- **VANILLA:** Consists of **M1**, which serves as a baseline 1-Block Layer Normalization (LN) Transformer architecture.
- **SIMPLE:** Comprises **M2** and **M3** for Simple Recursive Model, representing a 2-Block Root Mean Square (RMS) recursive structure configured with recursive steps of $T = 1$ and $T = 3$, respectively.
- **TRM:** Represents the Tiny Recursive Model, where **M4** denotes a 1-Block-RMS variant incorporating a latent variable (Z), and **M5** denotes a 1-Block-LN variant operating over joint latent configurations (Z, Y). $T = 5$ and $N = 3$ were used.
- **DEPTH:** Comprises Depth Recursive Model, where **M6** specifies a 1-Block-RMS design and **M7** represents a 2-Block-RMS architecture. The depth is adapted to $T = 16$ for both the Stress10K and Warpage10K dataset.

B. Datasets description

We evaluate recursive models across three datasets with partial DOE factorial cases, $N = factors^{levels}$—the Stress10k and Warpage10k datasets ($N = 5^4 \times 16 = 10{,}000$ pairs) generated by FEA simulation. Synthetic capacitor electrostatic field dataset generated by physics-informed neural network (PINN) ($N = 22^2 \times 15 = 7{,}265$ pairs).

Stress10k, Warpage10k [24]: Advanced packaging design requires understanding how material properties and geometric configurations influence thermo-mechanical behavior across spatial and depth-wise dimensions. Design variables –epoxy molding compound (EMC) coefficient of thermal expansion (CTE) and elastic modulus (E), die sizes ($Size_{dies}$), inter-die spacing (Gap_{dies}), and position within the package depth ($S^{(t)}$) –determine the resulting stress and warpage responses obtained from finite element analysis (FEA).

$$\mathbf{x_{stress}} = \left[\mathrm{EMC_{CTE}},\ \mathrm{EMC}_E,\ \mathrm{Size_{dies}},\ \mathrm{Gap_{dies}},\ S^{(t)}\right] \in \mathbb{R}^{5\times 1}$$
$$\mathbf{I_{stress}} = \left\{\mathbf{I}^{(1)}, \mathbf{I}^{(2)}, \ldots, \mathbf{I}^{(16)}\right\} \in \mathbb{R}^{13\times 13\times 16} \tag{6}$$

PINN [25]: The electrostatic field within an air-filled capacitor can be modeled by establishing collocation and boundary points across a two-dimensional grid. While the Laplace equation governs the internal collocation points, the system's boundary conditions define the perimeter. In this study, we implement five distinct boundary conditions (C1 through C5) in [25] governed by boundary parameters (a, b, d). Rather than treating all boundary metrics as static values, we allow the parameters to vary dynamically. Modifying these specific parameters significantly alters the resulting electrostatic field distribution, as demonstrated in Figure 1.

$$\mathbf{x}_{PINN} = \left[\mathrm{a}, \mathrm{b}, d^{(t)}\right] \in \mathbb{R}^{3\times 1}.$$
$$\mathbf{I}_{PINN} = \left\{\mathbf{I}^{(1)}, \mathbf{I}^{(2)}, \ldots, \mathbf{I}^{(15)}\right\} \in \mathbb{R}^{13\times 13\times 15} \tag{7}$$

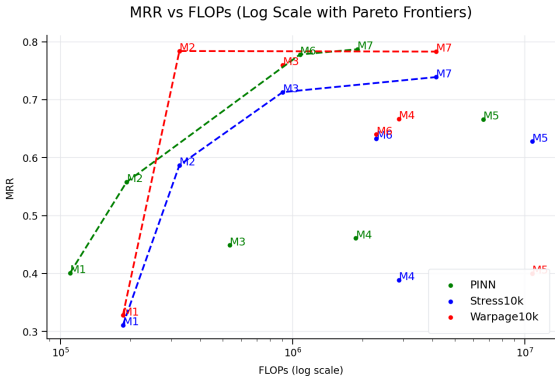

Fig. 4: Higher FLOPs means a GPU takes longer to run on a single request.

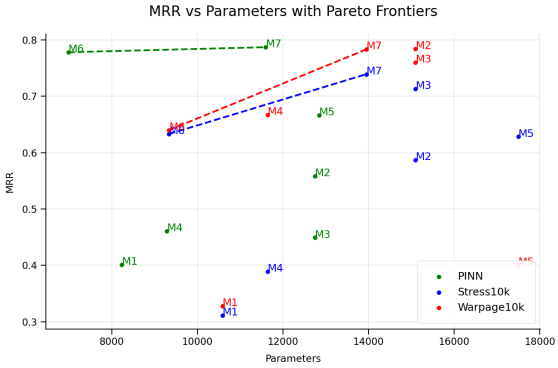

Fig. 5: Unlike FLOPs, we cannot "slowly" load a model into VRAM while running it; the entire model must fit into VRAM before inference begins.

C. Pareto Analysis

Figure 4 compares MRR against FLOPs across the three datasets. The Pareto frontiers differ across datasets: Stress10K progresses from M1→M2→M3→M7, indicating that M3 remains a competitive intermediate operating point; PINN follows M1→M2→M6→M7, where depth conditioning first becomes Pareto-optimal. Warpage10K transitions M2→M7, suggesting that the intermediate recursive variants are dominated. The TRM variants (M4–M5) do not lie on the Pareto frontier for any dataset, indicating a less favorable accuracy-to-computation trade-off. Figure 5 compares MRR against Parameters. Unlike the FLOP analysis, the Pareto frontiers consistently favor DEPTH. For PINN, Stress10K and Warpage10K, the Pareto frontier consists exclusively of the DEPTH models (M6–M7), where additional parameters yield further accuracy improvements while remaining Pareto-optimal. The TRM variants (M4–M5) remain well inside the Pareto region despite their larger parameter count, indicating that increasing parameter alone does not translate into better MRR.

IV. Conclusion

The proposed DEPTH (M6, M7) deliver the strongest or near-strongest retrieval accuracy while using among the fewest parameters and lowest FLOPs of all seven models

evaluated. The naive weight-sharing recursion of SIMPLE (M2,M3) is FLOP-efficient but accuracy-limited relative to M7 at comparable compute, while the dual-latent-state recursion of TRM (M4,M5) is both parameter- and FLOP-expensive without a commensurate accuracy return. These results support the central claim that explicitly uses depth as a recursively injected input in which each recursive output contributes an individual loss term during BPTT, is the primary factor behind the favorable accuracy-per-parameter and accuracy-per-FLOP trade-offs achieved for package-level Stress10k, Warpage10k, and PINN prediction.

REFERENCES

[1] S. Fan, H. Lu, S. Zhang, N. Cao, X. Zhang, and J. Li, "Graph-transformer-based surrogate model for accelerated converter circuit topology design," in *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC)*, 2024.

[2] Z. Li, K. Meidani, and A. B. Farimani, "Transformer for partial differential equations' operator learning," *Transactions on Machine Learning Research*, 2023.

[3] F. Zhang *et al.*, "Transformer self-attention encoder–decoder with multimodal deep learning for response time series forecasting and digital twin support in wind structural health monitoring," *Preprint*, 2026.

[4] S. Saha *et al.*, "Vision transformers on the edge: A comprehensive survey of model compression and acceleration strategies," *arXiv preprint arXiv:2503.02891*, 2025.

[5] C. Cioflan *et al.*, "Distributed inference with minimal off-chip traffic for transformers on low-power mcus," *arXiv preprint arXiv:2412.04372*, 2024.

[6] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, 2022.

[8] M. Dehghani *et al.*, "Universal transformers," in *International Conference on Learning Representations (ICLR)*, 2019.

[9] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[10] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning (ICML)*, 2020.

[11] Y. Rao *et al.*, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[12] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, and J. Hoffman, "Token merging: Your vit but faster," in *International Conference on Learning Representations (ICLR)*, 2023.

[13] N. Du *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," in *International Conference on Machine Learning (ICML)*, 2022.

[14] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations (ICLR)*, 2020.

[15] L. Yang *et al.*, "Looped transformers are better at learning learning algorithms," in *International Conference on Learning Representations (ICLR)*, 2024.

[16] Samsung SAIL Research, "Tiny recursive models," *arXiv preprint*, 2025.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning Workshop*, 2015.

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations (ICLR)*, 2022.

[19] T. Dao, "Flashattention-2: Faster attention with better parallelism," in *International Conference on Learning Representations (ICLR)*, 2024.

[20] NVIDIA, "Tensorrt-llm," GitHub Repository, 2023.

[21] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations (ICLR)*, 2022.

[22] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *International Conference on Learning Representations (ICLR)*, 2024.

[23] Mistral AI, "Mixtral of experts," *arXiv preprint*, 2024.

[24] K.-L. Lim and J. Lin, "Deep clustering based boundary-decoder net for inter and intra layer stress prediction of heterogeneous integrated ic chip," in *2025 IEEE 75th Electronic Components and Technology Conference (ECTC)*, 2025, pp. 239–245.

[25] K.-L. Lim, R. Dutta, and M. Rotaru, "Inverse prediction of capacitor multiphysics dynamic parameters using deep generative model," in *2023 IEEE 73rd Electronic Components and Technology Conference (ECTC)*, 2023, pp. 1825–1831.