RTL-3D: Timing-aware Tier Partitioning for 3D ICs Using Pre-synthesis Timing Analysis

Haoyang Xu[1], Zheng Yang[2], Zhen Zhuang[1*], Leilei Jin[1], Bei Yu[1], Sung Kyu Lim[3], Tsung-Yi Ho[1]
[1] The Chinese University of Hong Kong [2] Georgia Institute of Technology
[3] University of Southern California [*] Corresponding author

ABSTRACT

RTL-3D enables early-stage cross-tier timing analysis and optimization in 3D ICs. The framework utilizes a Transformer-based pre-synthesis timing analysis model to directly steer its core partitioning method, which features differentiable register tier partitioning. This creates a tight feedback loop between timing prediction and physical implementation, closing the loop for co-optimization. The experimental evaluation demonstrates that RTL-3D achieves significant improvements in sign-off timing metrics by a 59.3% reduction in WNS and a 75.4% reduction in TNS compared to state-of-the-art methods.

1 INTRODUCTION

State-of-the-art 3D physical design often relies on commercial 2D physical design tools [1]. However, this methodology struggles with the unique complexity of 3D structures, which intensifies the difficulty of timing optimization. To achieve accelerated timing closure, early and dedicated cross-tier timing analysis and optimization are essential for fully leveraging the performance potential of 3D ICs.

Tier partitioning, the critical step of assigning circuit elements across tiers in 3D ICs, directly determines the physical properties of cross-tier interconnections and thereby affects timing. In the pre-synthesis stage, the design retains maximal flexibility. A wide range of partitioning strategies can be applied to improve timing metrics. Therefore, optimizing the tier partitioning with efficient timing analysis during the pre-synthesis stage is necessary.

However, the primary challenge in realizing this potential is to accurately estimate the timing of different partitioning choices directly from RTL descriptions. Without effective timing information that reflects the 3D physical design, it is difficult to make informed partitioning decisions in the pre-synthesis stage. In the 2D IC domain, numerous studies have successfully employed machine learning (ML) techniques to enable pre-synthesis timing prediction [2–5]. However, their methods exclusively focus on 2D physical features, lack the capacity to capture critical 3D cross-tier characteristics, rendering them ineffective for accurate timing analysis in 3D ICs.

Another significant challenge is performing partitioning optimization amid incomplete information at the pre-synthesis stage. There are many explorations to optimize tier partitioning [6, 7]. However, these approaches are constrained by their dependence on detailed post-synthesis netlists and physical information, effectively restricting tier partitioning optimization to a late stage. Consequently, these approaches cannot leverage the flexibility offered in the pre-synthesis stage to make timing-driven partitioning decisions. As shown in Fig. 1, the previous tier partitioner cannot recognize the priority of registers, resulting in a partitioning scheme with unbalanced timing paths of Fig. 1(a). Due to efficient pre-synthesis timing analysis, we can make tier partitioning with better total negative slack (TNS) and worst negative slack (WNS) in Fig. 1(b).

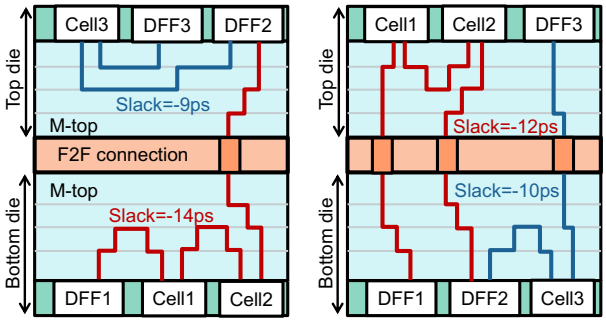

Figure 1: Comparison of different tier partitioning methods. (a) Tier partitioning without early stage timing guidance, resulting in WNS/TNS=-14/-23ps. (b) The timing-critical register can be specified in the pre-synthesis stage, optimizing tier partitioning result with WNS/TNS=-12/-22ps.

To address the aforementioned challenges, this paper proposes **RTL-3D**, the first framework that enables pre-synthesis cross-tier timing prediction for 3D IC partitioning using a Transformer-based approach. Built upon this predictive model, our method decomposes the tier partitioning problem into two distinct stages: register-level partitioning and combinational logic cell partitioning. Unlike previous works that either overlook or are incapable of early-stage planning, RTL-3D facilitates tier partitioning decisions directly at the RTL stage, long before physical implementation, to accelerate timing closure for 3D ICs. Our main contributions are listed below:

- We present RTL-3D, a 3D tier partitioning methodology that leverages pre-synthesis timing analysis to facilitate rapid timing closure in 3D ICs.
- We develop a differentiable register tier partitioning method for fast, pre-synthesis optimization. The partitioning method is powered by a Transformer-based timing model that delivers accurate path-level predictions, enabling seamless design closure.
- We integrate a backward net weight propagation methodology into the K-means-based logic cell partitioning algorithm, leveraging pre-synthesis timing information.
- Experimental results show that we achieve 59.3%/75.4% 3D full-chip WNS/TNS reduction over the SOTA tier partitioner TP-GNN [7], demonstrated on four industrial designs.

DAC '26, July 26–29, 2026, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07…$15.00
https://doi.org/10.1145/3770743.3804036

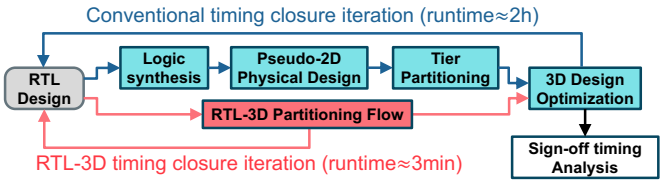

Figure 2: Using pre-synthesis timing analysis, the timing closure iterations can be greatly accelerated.

2 PRELIMINARIES

2.1 3D ICs Tier Partitioning

Most existing tier partitioning methods for 3D ICs utilize conventional algorithms or modern ML techniques. [8] employed simulated annealing to optimize wire-cost for cell-on-cell designs, leveraging high-density inter-die connections. [9] adopted a folding-based methodology using commercial 2D tools with full 3D backend analysis. More recently, TP-GNN [7] applied an unsupervised Graph Neural Network (GNN) to holistically comprehend technology and design parameters, showing significant quality improvements. While effective, these approaches fundamentally rely on post-synthesis netlists, confining optimization to a late stage where architectural flexibility is substantially reduced, thereby missing the opportunity for early-stage timing-driven tier partitioning strategy analysis.

2.2 Pre-synthesis Timing Analysis

The significant gap between early design representations and sign-off results makes ML well-suited for pre-synthesis timing prediction. The key challenges are extracting meaningful information from HDL code and transforming it into ML-suitable features. For the first challenge, graph-based representations like SOG [3, 5] and BOG [4] have emerged as effective solutions. These graphs, constructed through pseudo logic synthesis, create structural netlist analogs that are more amenable to ML. However, the intricate structure of 3D ICs introduces substantial timing prediction challenges, precluding the direct application of 2D predictors.

2.3 Tier Partitioning with Pre-synthesis Timing

Pre-synthesis timing analysis provides essential timing guidance, enabling the identification of critical paths and registers, which directly inform and optimize the partitioning strategy for improved sign-off timing performance. Therefore, it is crucial to break through the timing bottleneck of conventional 3D design flows. On the other hand, performing this timing-driven partitioning early in the design flow is highly efficient. It allows for rapid design space exploration and significantly accelerates the timing closure loop, all while the design architecture retains maximum flexibility for substantial modifications. As shown in Fig. 2, RTL-3D enables designers obtain the sign-off timing directly from tier partitioning but not after 3D placement and routing. This combination of performance improvement and accelerated closure is unattainable at later, more constrained design stages.

3 METHODOLOGY

3.1 Overview

Pseudo-3D methodologies [1], in contrast to academic true 3D approaches [11, 12], construct 3D ICs by leveraging commercial 2D design tools. This strategy ensures the resulting designs meet commercial standards for power, performance, and area (PPA) and are

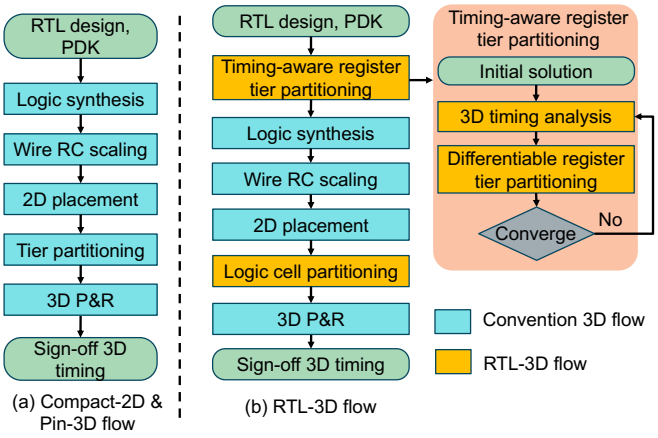

Figure 3: SOTA pseudo 3D physical design flow Compact-2D [6] & Pin-3D [10] vs. our timing-aware partitioning flow RTL-3D, which introduces additional pre-synthesis register tier partitioning and post-synthesis logic cell partitioning.

manufacturing-ready in the GDSII format. The core principle involves using these tools to determine the optimal (x, y) placement for standard cells, while the z-coordinate is decided through explicit tier assignments. Fig. 3(a) illustrates a conventional flow integrating the Compact-2D design [6] with the Pin-3D optimizer [10]. Based on this design flow, we propose RTL-3D, a timing-aware tier partitioning framework designed for versatile integration into existing 3D physical design methodologies. As shown in Fig. 3(b), the RTL-3D components (highlighted in orange) augment the baseline through pre-synthesis timing-aware register tier partitioning and post-synthesis logic cell tier partitioning.

The complete RTL-3D workflow is depicted in Fig. 4, which comprises four main stages: 1) Pre-synthesis processing for feature extraction from the RTL design and PDK; 2) 3D timing analysis, which employs a Transformer-based ML model to predict the sign-off timing metrics of a given register tier partitioning scheme; 3) Differentiable register tier partitioning to optimize the tier assignment of registers; and 4) Post-synthesis logic cell partitioning, which partitions the remaining non-register cells after 2D placement.

3.2 Pre-synthesis 3D Timing Analysis

Accurate pre-synthesis timing prediction is challenging due to the difficulty in extracting predictive features directly from RTL code. To bridge this abstraction gap, we employ the Simple Operator Graph (SOG) [4], which constructs a netlist-like graph through pseudo logic synthesis while preserving structural fidelity and offering significant speed advantages over conventional logic synthesis [3].

Since timing analysis fundamentally relies on timing paths, and the SOG provides a structural representation amenable to such analysis, we extract register-to-register paths from the SOG that resemble critical paths in the sign-off layout. Specifically, for each register in the SOG, we extract the K worst timing paths using the STA tool. The selection of K involves a critical trade-off: excessively large K values introduce noise, while insufficiently small K values result in loss of optimal training data. To determine the optimal K value, we consider the startpoint matching ratio between the SOG and the layout. Fig. 5(a) illustrates a matching example where $K < 3$ fails to capture critical register R3. Fig. 5(b) shows the matching ratio

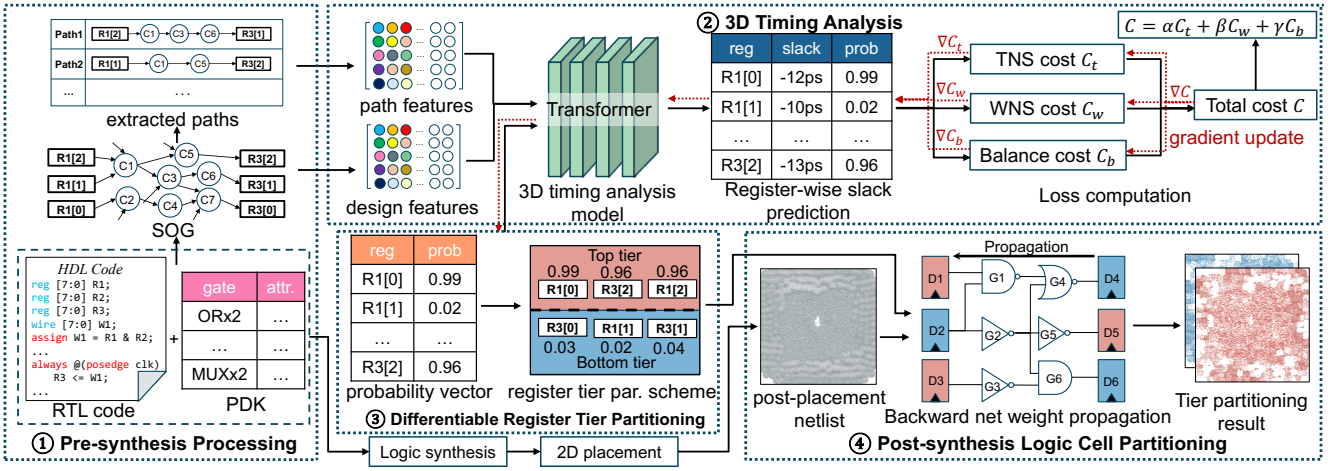

Figure 4: The process of RTL-3D. The flow begins with SOG construction. The 3D timing analysis model predicts the register-wise slack by design feature and path feature from the constructed SOG for a given register tier partitioning scheme. Considering TNS, WNS, and tier balance, we adopt gradient descent to find the optimal register tier partitioning result. By backward net weight propagation, we apply weighted min-cut on the 2D design to partition the rest of the logic cells.

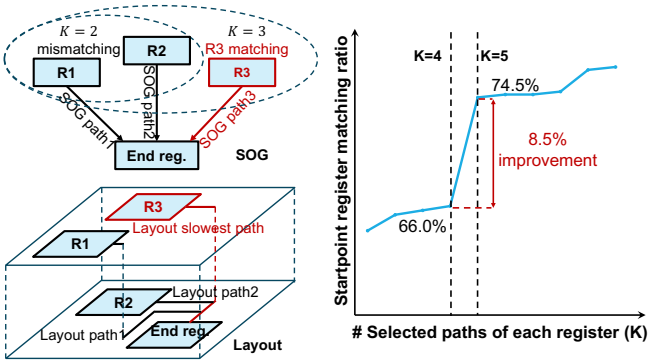

Figure 5: Correlation of startpoint register matching ratio and K. (a) An example of startpoint matching between SOG and layout. Matching fails when $K < 3$, while $K = 3$ succeeds. (b) The growth of the startpoint matching ratio as K increases. Note that there is an 8.5% improvement from $K = 4$ to $K = 5$.

Table 1: Summary of features used in RTL-3D.

Type	Feature	Description
Design	frequency	Design frequency
	num regs	Number of registers
	num cells	Number of SOG cells
Path	AT SOG	Path arrival time on SOG
	path level	Path level
	type cells	Numbers of each type of gates
	fanout	Statistics of fanout
	cap	Statistics of capacitance
3D	tier reg	z-location of register
	tier same	Whether start and end reg in same tier

between SOG paths and layout paths across varying K values on the AES design. Our analysis reveals diminishing improvements beyond $K = 5$, justifying our selection of $K = 5$ for optimal performance with a 74.5% matching ratio.

However, the SOG remains essentially a 2D graph. We therefore augment our feature set by extracting 3D characteristics from Compact-2D [6] tier partitioning results to enable more precise predictions. Table 1 details the complete set of input features, categorized into: design features from RTL or SOG, path features after SOG path selection, and 3D features from tier partitioning results. The integration of 3D-specific features enables the model to capture the unique timing characteristics of vertical interconnects. The corresponding labels are derived from sign-off register-level slack values. We adopt a Transformer architecture [13] for its superior capability in capturing complex feature interactions. The self-attention mechanism explicitly models relationships between heterogeneous features, including 2D structural characteristics and 3D physical features, enabling the model to learn how these factors collectively influence timing.

Formally, our timing analysis model is defined as $\hat{S} = F(P, G)$, where $P = \{p_i | 1 \le i \le |P|,\ p_i \in \{0,\ 1\}\}$ represents the input register tier partitioning scheme, G is the SOG generated from RTL code, and $\hat{S} = \{\hat{s}_i | 1 \le i \le |\hat{S}|\}$ denotes the predicted register-wise slack.

3.3 Differentiable Register Tier Partitioning

The development of our timing analysis model enables pre-synthesis timing prediction, but effectively utilizing this predictive capability for tier partitioning presents significant algorithmic challenges. Conventional partitioners operate post-synthesis, where the complete netlist and accurate timing information are available. However, at the pre-synthesis stage, we lack the final netlist and must make critical tier assignment decisions based on the structurally different SOG representation. To overcome this challenge, we reformulate register tier partitioning as a continuous optimization problem, leveraging the differentiability of our timing model. The core insight is that by relaxing discrete tier assignments to continuous probabilities, we can employ gradient-based optimization to efficiently navigate the solution space while considering complex timing objectives.

Formally, given the SOG G of a design, we seek the optimal register z-location assignment P^* that minimizes a comprehensive cost function $C(P)$, which integrates three essential objectives through a weighted combination:

$$C(P) = \alpha \cdot C_t(P) + \beta \cdot C_w(P) + \gamma \cdot C_b(P). \quad (1)$$

Smooth TNS cost approximation. The TNS component $C_t(P)$ provides a differentiable approximation of the traditional TNS metric, which sums the magnitudes of all negative slack violations. We formulate this using a smooth logarithmic barrier function:

$$C_t(P) = -\frac{1}{\tau} \sum_{i=1}^{|P|} \ln(\exp(-\tau \cdot \hat{s}_i) + 1). \quad (2)$$

For registers with positive slack ($\hat{s}_i > 0$), the term $\exp(-\tau \cdot \hat{s}_i)$ approaches zero, making $C_t(P)$ negligible. Conversely, for registers with negative slack ($\hat{s}_i < 0$), the term dominates and $C_t(P)$ approximates $\sum \max(0, -\hat{s}_i)$. The temperature parameter τ controls the sharpness of this smooth approximation. Mathematically, as $\tau \to \infty$, the approximation converges to the exact TNS function, while smaller values of τ provide smoother gradients that facilitate stable gradient-based optimization. We select τ through cross-validation to balance approximation accuracy and convergence properties.

Differentiable worst negative slack formulation. The WNS component $C_w(P)$ approximates the maximum timing violation through a log-sum-exp formulation:

$$C_w(P) = -\frac{1}{\tau} \ln \sum_{i=1}^{|P|} \exp(-\tau \cdot \hat{s}_i). \quad (3)$$

The smooth maximum operation effectively captures the worst-case timing scenario while maintaining differentiability. As $\tau \to \infty, C_w(P)$ converges to $-\max_i \hat{s}_i$, accurately representing the traditional WNS metric. This formulation ensures that registers with the most negative slacks receive the highest gradient magnitudes, directing the optimization to prioritize critical timing paths.

Balance-aware regularization. The balance component $C_b(P)$ enforces equitable distribution of registers across tiers through a quadratic penalty term:

$$C_b(P) = \left(\frac{\sum_{i=1}^{|P|} p_i}{|P|} - 0.5 \right)^2. \quad (4)$$

The balance term prevents degenerate solutions where all registers are assigned to a single tier, which could occur if only timing objectives were considered.

Gradient-based optimization. The complete gradient of the cost function with respect to tier assignment probabilities is computed through the chain rule:

$$\nabla_P C(P) = \alpha \cdot \nabla_P C_t(P) + \beta \cdot \nabla_P C_w(P) + \gamma \cdot \nabla_P C_b(P), \quad (5)$$

where the gradients incorporate the timing model's Jacobian $J_F(P)$:

$$\nabla_P C_t(P) = \left[\frac{\exp(-\tau \cdot \hat{s}_i)}{1 + \exp(-\tau \cdot \hat{s}_i)} \right]_{i=1}^{|P|} \cdot J_F(P), \quad (6)$$

$$\nabla_P C_w(P) = \left[\frac{\exp(-\tau \cdot \hat{s}_i)}{\sum_j \exp(-\tau \cdot \hat{s}_j)} \right]_{i=1}^{|P|} \cdot J_F(P), \quad (7)$$

$$\nabla_P C_b(P) = \frac{2}{|P|^2} \left(\sum p_i - \frac{|P|}{2} \right). \quad (8)$$

The Jacobian matrix $J_F(P)$ captures the sensitivity of timing predictions to tier assignment changes, which is efficiently computed via automatic differentiation. This enables our optimization to intelligently navigate the solution space by understanding how register placements affect overall timing.

Algorithm 1: Differentiable register tier partitioning

Input: SOG G, timing model F
Output: Optimal tier assignment P^*

Initialize $P \sim \text{Uniform}(0,1)^N$;
Set learning rate η, convergence threshold ϵ;
repeat
  $\hat{S} \leftarrow F(P, G)$;
  $C_t \leftarrow -\frac{1}{\tau} \sum_{i=1}^{N} \ln(\exp(-\tau \cdot \hat{s}_i) + 1)$;
  $C_w \leftarrow -\frac{1}{\tau} \ln \sum_{i=1}^{N} \exp(-\tau \cdot \hat{s}_i)$;
  $C_b \leftarrow \left(\frac{\sum_{i=1}^{N} p_i}{N} - 0.5 \right)^2$;
  $C(P) \leftarrow \alpha \cdot C_t + \beta \cdot C_w + \gamma \cdot C_b$;
  Compute $\nabla_P C(P)$ via automatic differentiation;
  $P \leftarrow P - \eta \cdot \nabla_P C(P)$;
until $\|\nabla_P C(P)\|_2 < \epsilon$ *or maximum iterations reached*;
$P^* \leftarrow \{\mathbb{I}[p_i > 0.5] \mid i = 1, \ldots, N\}$;
return P^*;

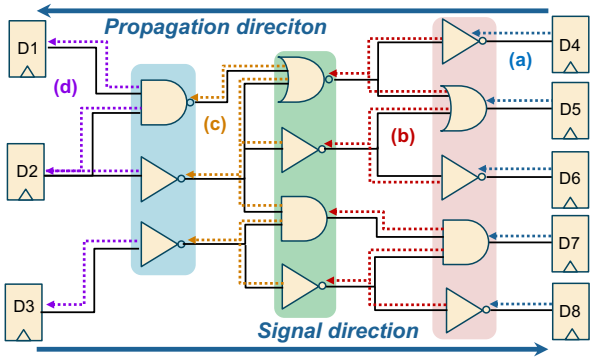

Figure 6: The backward net weight propagation process. Net weights are computed following the order of (a), (b), (c), (d).

We relax the discrete tier assignment $p_i \in \{0, 1\}$ to continuous probabilities $p_i \in [0, 1]$, transforming the combinatorial optimization into a continuous problem amenable to gradient methods. As illustrated in Fig. 4, our flow begins with RTL code and PDK, constructing the SOG for 2D feature extraction. For new designs without ground-truth tier partitioning, we initialize P_0 with random values in $[0, 1]$. Algorithm 1 formalizes our differentiable partitioning approach.

3.4 Post-synthesis Logic Cell Partitioning

Following the pre-synthesis register tier partitioning, we proceed to determine the z-location of logic cells based on the established register partitioning and corresponding timing information. A fundamental observation guiding our approach is that for timing-critical nets, maintaining 2D connectivity is generally preferable to 3D inter-tier connections, as the latter introduce significant timing overhead due to inter-die connection delays.

Unlike the bin-based min-cut algorithm employed in Compact-2D [6], we introduce a net weighting strategy that incorporates both register timing criticality and tier assignment information shown in Fig. 4. To evaluate the timing importance of registers, the register priority q_r is computed as

$$q_r = \exp\left(-\frac{\hat{s}_r - \hat{s}_{\min}}{\hat{s}_{\max} - \hat{s}_{\min}}\right), \quad (9)$$

where $\hat{s}_{\min}$ and $\hat{s}_{\max}$ represent the minimum and maximum slack values across all registers. This formulation ensures that registers with worse slack values receive exponentially higher priorities while maintaining normalized scaling across different designs.

To efficiently compute net weights that incorporate tier partitioning awareness, we employ a layered dynamic programming approach. The weight propagation employs a bitmask-based approach in which each register is mapped to a unique bit position. The algorithm is initialized by setting bits corresponding to directly connected registers in each net's bitmask. The core propagation operates through iterative frontier expansion: at each level, net bitmasks are aggregated to fan-in cells, then redistributed to nets of the next level. This iteration flow ensures comprehensive reachability analysis while maintaining computational efficiency through bit-level operations.

As illustrated in Fig. 6, the propagation follows a structured sequence from (a) to (d). Phase 1 (net-to-cell aggregation) accumulates reachability information at intermediate logic cells, like nets from (a) to cells in red. Phase 2 (cell-to-net propagation) disseminates this information to the nets of the next level, like gates in red to nets in (b). The process iterates until no new register bits can be propagated, ensuring all reachable register dependencies of (a), (b), (c), and (d) are captured in the final net weights.

The final net weight w_n is computed directly from its bitmask b_n and the tier assignment P^* using:

$$w_n = \left(1 + \alpha \cdot \frac{\sum_{i \in bit(b_n)} \mathbb{I}[p_i^* = \text{mode}(b_n)]}{\text{bitcount}(b_n)}\right) \cdot \sum_{i \in bit(b_n)} q_i, \quad (10)$$

where $\text{bit}(b_n)$ indicates that the set of registers contained in b_n, $\text{mode}(\cdot)$ denotes the most frequent tier assignment among reachable its bitmask, and $\text{bitcount}(*)$ is the number of bits. This formulation rewards nets whose reachable registers predominantly belong to the same tier, as such configurations minimize inter-tier connections and associated timing penalties.

With the computed net weights, we perform logic cell partitioning using a two-stage approach. Since bin-based layout partitioning is cell-independent and lacks adaptive clustering capability, we first apply K-means clustering to group cells based on spatial proximity and connectivity patterns. The value of clusters is decided by the design complexity to ensure a stable partitioning performance. Subsequently, within each cluster, we employ Fiduccia-Mattheyses min-cut [14] partitioning that incorporates the net weights as cutting costs. Different from the directly applied min-cut algorithm, our weighted min-cut focuses on timing criticality, but not the inter-die connections. Finally, all logic cells are efficiently partitioned into 2 tiers to finish the tier partitioning.

4 EXPERIMENTAL RESULTS

We evaluate the RTL-3D flow using ground truth data generated by state-of-the-art pseudo-2D and 3D design flows, Compact-2D (C2D) [6] and Pin-3D [10], on four industrial designs: LDPC, AES, VGA, and JPEG. The SOG is constructed following the method in [4]. All benchmarks are synthesized with an in-house predictive 3nm technology node using Synopsys Design Compiler [15], assuming face-to-face 3D hybrid bonding. We utilize Synopsys ICC2 [16] for pseudo-2D and 3D placement and routing, while sign-off timing analysis is assessed using Synopsys PrimeTime SI [17] with back-annotated parasitics. The Transformer model and tier partitioning are implemented in Python with PyTorch and PyTorch Geometric.

Table 2: Model performance comparison between RTL-timer [4] and RTL-3D on sign-off register-wise slack, TNS, and WNS.

Metric	Method	R	R^2	MAPE (%)	RRSE
Register slack	RTL-timer [4]	0.88	0.77	10.2	0.42
	RTL-3D	**0.94**	**0.89**	**8.17**	**0.38**
TNS	RTL-timer [4]	0.85	0.72	56.3	0.11
	RTL-3D	**0.98**	**0.96**	**13.2**	**0.05**
WNS	RTL-timer [4]	0.61	0.37	66.9	0.64
	RTL-3D	**0.96**	**0.92**	**12.9**	**0.56**

Table 3: Detailed comparisons of sign-off 3D timing results between Compact-2D [6], the state-of-the-art tier partitioning flow TP-GNN [7], and our RTL-3D over four industrial benchmarks in an in-house predictive 3nm technology node. Fmax denotes the maximum effective frequency. Note that all flows are run with Pin-3D [10] optimizer for 3D optimization. The best performance in each column is colored in green.

Method	WNS (ps)	TNS (ps)	Fmax (GHz)	#Timing violations	Wirelength (m)	Partition time (s)
AES (3.3GHz, #cells: 119354, register area ratio: 33.2%)						
C2D [6]	-5.09	-13.63	2.98	6	0.258	90
TP-GNN [7]	-2.75	-3.42	3.27	3	**0.246**	1006
RTL-3D	**0.22**	**0**	**3.30**	**0**	0.247	**27**
LDPC (2.5GHz, #cells: 52727, register area ratio: 12.7%)						
C2D [6]	-23.8	-202.44	2.02	42	0.174	24
TP-GNN [7]	-5.76	-47.90	2.36	20	0.176	261
RTL-3D	**-3.04**	**-9.04**	**2.42**	**6**	**0.173**	**14**
VGA (3.3GHz, #cells: 52122, register area ratio: 57.8%)						
C2D [6]	-6.51	-95.55	3.07	38	0.095	**27**
TP-GNN [7]	-6.87	-75.07	3.06	34	0.095	524
RTL-3D	**-2.60**	**-4.08**	**3.21**	**8**	**0.084**	115
JPEG (2.5GHz, #cells: 325336, register area ratio: 37.6%)						
C2D [6]	-36.71	-857.10	1.93	78	**0.588**	716
TP-GNN [7]	-5.05	-18.58	2.38	7	0.590	3826
RTL-3D	**-4.05**	**-13.77**	**2.40**	**7**	0.591	**163**

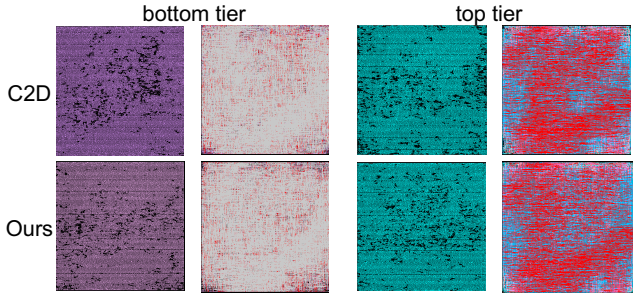

Figure 7: Placement and routing layout of Compact-2D vs. ours on LDPC@2.5GHz.

4.1 Pre-synthesis Timing Prediction Accuracy

To train the register-tier partitioning timing analysis model described in Section 3.2, we build the dataset using C2D tier partitioning results, with 20% reserved for testing. Each design is synthesized at

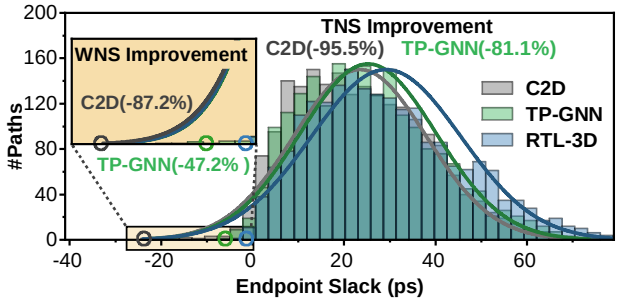

Figure 8: The slack distribution comparison of Compact-2D [6], TP-GNN [7] and RTL-3D on LDPC benchmark.

Table 4: Comparison of WNS and TNS between RTL-3D with pre-synthesis register tier partitioning and RTL-3D with post-synthesis tier partitioning. Logic cells are partitioned after 2D placement in both flows.

Design	Pre-synthesis reg. tier par.		Post-synthesis reg. tier par.	
	WNS (ps)	TNS (ps)	WNS (ps)	TNS (ps)
AES	**0.22**	**0.00**	-1.86	-2.35
LDPC	**-3.04**	**-9.04**	-6.82	-33.60
VGA	**-2.60**	**-4.08**	-2.72	-5.07
JPEG	**-4.05**	**-13.77**	-8.83	-18.57
Avg.	**0.38**	**0.45**	1.00	1.00

Step	Runtime
SOG Construction	2.3%
Register Tier Partitioning	0.1%
3D Timing Analysis	<0.01%
Logic Cell Partitioning	3.9%

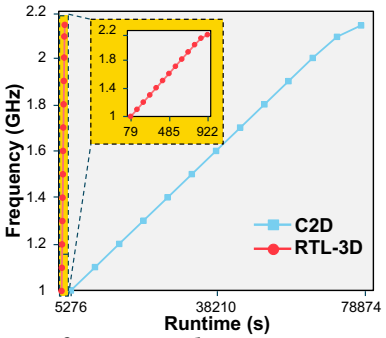

Figure 9: Runtime breakdown of RTL-3D with runtime comparison of timing closure between Compact-2D [6] and RTL-3D. Runtime is listed as the proportion of the full 3D design flow.

three different frequencies to enhance the model's generalizability. Multiple metrics are employed to fully evaluate our 3D timing analyzer: correlation coefficient (R), coefficient of determination (R^2), mean absolute percentage error (MAPE), and root relative squared error (RRSE). We compare our timing analysis model with RTL-timer [4], a SOTA 2D RTL timing prediction flow. For fair comparison, we train RTL-timer [4] on the same designs and PDK. Note that our timing predictions are derived from register tier partitioning result without ground-truth tier information. As shown in Table 2, RTL-3D achieves higher correlation coefficients of 0.98 for TNS and 0.96 for WNS compared to RTL-timer [4]. Additionally, it attains MAPE values of 13.2% for TNS and 12.9% for WNS, with a lower RRSE.

4.2 Tier Partitioning Performance

We validate RTL-3D on four industrial designs, LDPC, JPEG, AES, and VGA, comparing it with C2D and TP-GNN [7], a SOTA GNN-based tier partitioning method. These benchmarks cover a wide range of design complexities and register area ratios, comprehensively demonstrating our method's effectiveness across different design types. Detailed results are presented in Table 3. Notably, RTL-3D consistently outperforms both C2D and TP-GNN regarding timing metrics across all designs. On average, it achieves improvements of 59.3% in WNS and 75.4% in TNS savings compared to TP-GNN. Consequently, the RTL-3D flow enables higher effective frequencies and fewer timing violations than both C2D and TP-GNN. The runtime of RTL-3D partitioning is calculated by post-synthesis logic cell partitioning, since register tier partitioning can be performed simultaneously with logic synthesis in RTL-3D flow, and logic synthesis is much slower than register tier partitioning. From Table 3, RTL-3D finishes tier partitioning faster than C2D and TP-GNN on most designs, which shows the efficiency of our method. Fig. 8 further displays the slack distribution comparison. It is noteworthy that, compared with TP-GNN and C2D, the peak of RTL-3D slack distribution moves toward the positive slack direction, showcasing the benefit of timing-aware tier partitioning enabled by timing prediction. Fig. 7 illustrates the post-placement and post-routing layouts for both tiers of C2D and RTL-3D to show the practicality of our method.

4.3 Ablation Study

Previous partitioning methods adopt post-synthesis tier partitioning. To quantify the benefit of our novel pre-synthesis register tier partitioning, we compare against a baseline where registers are partitioned post-synthesis alongside logic cells. As shown in Table 4, our approach achieves significant timing improvements, with average reductions of 62% in WNS and 55% in TNS across all benchmarks. This demonstrates that early register positioning unlocks greater timing optimization potential by guiding subsequent synthesis and placement stages, rather than being constrained by a fixed post-synthesis netlist structure.

4.4 Benefits of Pre-synthesis Timing Prediction

Beyond improved timing quality, our pre-synthesis timing prediction fundamentally accelerates design timing convergence. Conventional flows detect violations only after full layout implementation, necessitating costly iterations. Fig. 9 illustrate the timing closure runtime comparison to C2D, RTL-3D completes iterations about 80× faster than the conventional C2D flow by terminating after register-tier timing analysis, before time-consuming logic synthesis and physical implementation.

5 CONCLUSIONS

In this paper, we propose RTL-3D, a novel two-phase tier partitioning framework based on the pre-synthesis timing analysis. First, we introduce a differentiable register tier partitioning algorithm using the transformer-based 3D timing analysis engine. Then, we fully utilize the timing register tier partitioning result and register-level timing information to partition the logic cells into 2 tiers. Experimental results show that RTL-3D not only optimizes sign-off timing but also accelerates the design loop.

Acknowledgments

This research was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government. This work is jointly supported by the Research Grants Council of Hong Kong SAR No. CUHK14211324 and No. T46-415/25-R.

REFERENCES

[1] H. Park, B. W. Ku, K. Chang, D. E. Shim, and S. K. Lim, "Pseudo-3D Approaches for Commercial-Grade RTL-to-GDS Tool Flow Targeting Monolithic 3D ICs," in *ACM International Symposium on Physical Design (ISPD)*, 2020, p. 47–54.

[2] P. Sengupta, A. Tyagi, Y. Chen, and J. Hu, "How Good Is Your Verilog RTL Code? A Quick Answer from Machine Learning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[3] W. Fang, Y. Lu, S. Liu, Q. Zhang, C. Xu, L. W. Wills, H. Zhang, and Z. Xie, "MasterRTL: A Pre-Synthesis PPA Estimation Framework for Any RTL Design," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023, pp. 1–9.

[4] W. Fang, S. Liu, H. Zhang, and Z. Xie, "Annotating Slack Directly on Your Verilog: Fine-Grained RTL Timing Evaluation for Early Optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2024.

[5] M. Wang, Y. Wen, B. Sun, J. Mu, J. Li, X. Wang, J. J. Ye, B. Yu, and H. Li, "Bridging Layout and RTL: Knowledge Distillation based Timing Prediction," in *International Conference on Machine Learning (ICML)*, 2025.

[6] B. W. Ku, K. Chang, and S. K. Lim, "Compact-2D: A Physical Design Methodology to Build Commercial-Quality Face-to-Face-Bonded 3D ICs," in *ACM International Symposium on Physical Design (ISPD)*, 2018, p. 90–97.

[7] Y.-C. Lu, S. S. Kiran Pentapati, L. Zhu, K. Samadi, and S. K. Lim, "TP-GNN: A Graph Neural Network Framework for Tier Partitioning in Monolithic 3D ICs," in *ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[8] G. Berhault, M. Brocard, S. Thuries, F. Galea, and L. Zaourar, "3DIP: An iterative partitioning tool for monolithic 3D IC," in *IEEE International 3D Systems Integration Conference*, 2016, pp. 1–5.

[9] O. Billoint, H. Sarhan, I. Rayane, M. Vinet, P. Batude, C. Fenouillet-Beranger, O. Rozeau, G. Cibrario, F. Deprat, A. Fustier, J.-E. Michallet, O. Faynot, O. Turkyilmaz, J.-F. Christmann, S. Thuries, and F. Clermidy, "A comprehensive study of Monolithic 3D cell on cell design using commercial 2D tool," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2015, pp. 1192–1196.

[10] S. S. K. Pentapati, K. Chang, V. Gerousis, R. Sengupta, and S. K. Lim, "Pin-3D: A Physical Synthesis and Post-layout Optimization Flow for Heterogeneous Monolithic 3D ICs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020.

[11] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, "TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 4, pp. 497–509, 2013.

[12] X. Zhao, S. Chen, Y. Qiu, J. Li, Z. Huang, B. Xie, X. Li, and Y. Bao, "iPL-3D: A Novel Bilevel Programming Model for Die-to-Die Placement," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017, p. 6000–6010.

[14] C. Fiduccia and R. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions," in *ACM/IEEE Design Automation Conference (DAC)*, 1982, pp. 175–181.

[15] Synopsys,"Design Compiler User Guide", https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[16] Synopsys, "IC Compiler II User Guide", https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html.

[17] Synopsys, "Primetime User Guide", https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html.