RTL-Sequencer: Towards Scalable RTL Timing Prediction with the Sequence-based Paradigm

Ziyan Guo, Wenji Fang, Wenkai Li, Yuchao Wu, Shang Liu, Zhiyao Xie*
Hong Kong University of Science and Technology (HKUST)
{zguoby, wfang838, wlidm, ywu092, sliudx}@connect.ust.hk, eezhiyao@ust.hk

ABSTRACT

Accurate timing prediction at the register-transfer level (RTL) is a longstanding challenge in design automation. Existing graph-based methods struggle with limited receptive fields, high complexity, and a lack of signal directionality. We present RTL-Sequencer, a novel sequence-based paradigm that enables scalable RTL timing prediction via linearizing logic cones by breadth-first traversal and applying modern linear sequence models. Furthermore, sequence models are customized by four synergistic techniques, including sequence shuffling, bidirectional modeling, differentiable modeling, and a hybrid graph-sequence architecture. Extensive experiments demonstrate significant improvements of RTL-Sequencer over state-of-the-art baselines, advancing early-stage timing optimization.

1 INTRODUCTION

Timing remains a critical optimization objective in the design of digital integrated circuits. A fundamental limitation in the conventional design methodologies is that accurate timing analysis is only available after the post-layout stage, due to the lack of accurate physical information like wire length information [16]. Consequently, this delayed feedback undermines the efficacy of timing-driven optimizations during early design phases, including register-transfer level (RTL) design, thereby significantly extending design iterations and increasing time-to-market.

To bridge this gap, Machine Learning (ML) has emerged as a promising data-driven paradigm for early-stage timing prediction. As shown in Table 1, prior research has extensively investigated ML-based timing prediction across various design stages, including the layout, the netlist, and the most challenging RTL stage due to its high level of abstraction [9, 11, 32]. At the RTL stage, Hardware Description Language (HDL) code is typically transformed into an intermediate representation called the Boolean Operator Graph (BOG) [9, 32], which is a directed graph of Boolean logic gates. ML–based timing prediction is then applied to logic cones, defined as subgraphs of the BOG that encompass an output register together with its all fan-in input registers and combinational nodes.

However, we observe that these cones are generated before synthesis and consequently exhibit deep, large-scale topological structures, often comprising more than ten thousand nodes with maximum depths exceeding one hundred. Such complexity presents substantial challenges for ML-based methodologies to accurately model long-range timing dependencies within the cones. As summarized in Table 1, these methodologies can be broadly classified

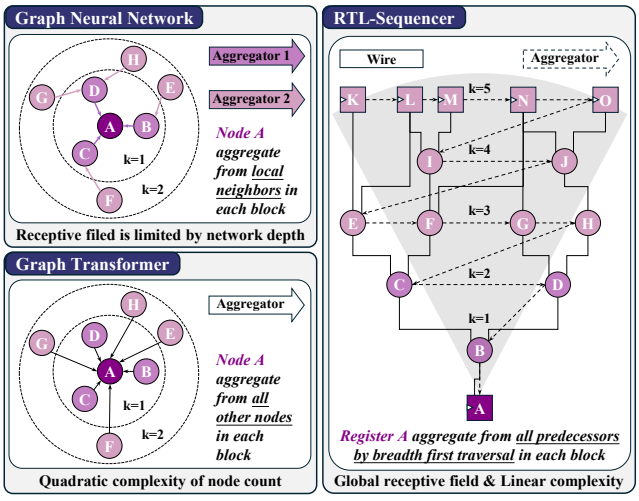


Figure 1: RTL-Sequencer vs. conventional graph-based approaches, such as the Graph Neural Network and the Graph Transformer, highlighting the scalability of the sequence-based paradigm on the receptive field and the node capacity.

into four categories: **(i) Traditional ML**, such as Random Forest [2] and XGBoost [5], primarily model statistical characteristics of logic cones, thereby neglecting their topological structure that is critical for accurate timing analysis. **(ii) GNNs including Graph Convolutional Networks (GCNs) [20] and Graph Attention Networks (GAT) [31]** suffer from constrained receptive fields, which are inherently bounded by network depth [13, 25]. Such restrictions hinder their applicability to pre-synthesis cones with substantial logic depth. Attempts to mitigate this issue by simply stacking additional GNN layers often exacerbate the problem of over-smoothing [26], resulting in homogenized node embeddings that lack discriminative representation. Consequently, the generalization capacity of these models deteriorates significantly when applied to large-scale designs. **(iii) GNN-Propagation** or DAGNN [29] aims to provide a global receptive field by computing representations layer by layer within the cone. Nevertheless, this architecture imposes strict inter-layer dependencies, resulting in training quadratic complexity with cone depth. Such complexity not only impedes training parallelization but also exacerbates error accumulation in deeper layers, thereby constraining scalability to deeply structured designs. **(iv) Graph Transformers (GTs)** [42] offer global receptive fields; however, they suffer from quadratic complexity with node count and over-emphasize node relationships rather than signal directionality. In addition, TF-Predictor [3] incorporates a Transformer as a sequence model, but it only represents specified paths at the relatively tractable layout stage and incurs quadratic node complexity.

To address the inherent limitations of graph-based methodologies, we introduce a novel framework, **RTL-Sequencer**, which

*Corresponding Author

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

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804056

Stage	Type	Works	Global Recept.	Linear Complex.
Layout	Traditional	[1, 15, 18, 19, 22]	–	–
	GNN-Prop.	[6, 14, 17, 34, 43]	✓	×
	GT	[4, 44]	✓	×
Netlist	Traditional	[39]	–	–
	GNN	[36, 37]	×	✓
RTL	Traditional	[9, 11, 27]	–	–
	GNN	[23]	×	✓
	GNN-Prop.	[32, 33]	✓	×
	GT	[8, 10, 38]	✓	×
	Sequencer	Ours	✓	✓

Table 1: Summary of existing methods for timing prediction.

establishes a scalable sequence-based paradigm for the challenging RTL timing prediction. **Our key insight** lies in customizing linear sequence models [7, 12, 24, 28, 40] instead of graph-based models. As shown in Fig. 1, such a paradigm shift is able to offer several **critical advantages over graph-based models: (i)** Global receptive fields for long-range timing dependencies independent of network depth; **(ii)** Linear computational complexity with respect to node count and cone depth; **(iii)** Explicit preservation of signal directionality, thus mitigating over-smoothing; **(iv)** Elimination of strict intra-layer dependencies. To achieve this sequence-based paradigm, an inverse Breadth-First Traversal (BFT) from the output register provides a basic solution to convert cones into sequences.

However, the straightforward adoption of deterministic BFT and conventional sequence models can be sub-optimal, primarily due to the loss of topological information when topology-invariant cones are reduced to fixed, unidirectional, breadth-first node sequences. To address these limitations, we propose **four synergistic techniques to customize sequence models further** to this challenging task: **(i) Sequence Shuffling** to expose sequence models to diverse sequence variants of cones rather than the fixed node ordering introduced by deterministic BFT. Specifically, nodes within each BFT depth are randomly permuted during training as data augmentation, while preserving logical dependencies and timing closure, thereby improving generalization across heterogeneous RTL designs. **(ii) Bidirectional Sequence Modeling** to capture the bi-directional flow of signal propagation in logic cones, therefore enabling deeper network blocks, via the representation awareness of later nodes to the earlier nodes. **(iii) Differentiable Sequence Modeling** to ensure that node representations align with depth-first critical timing paths rather than breadth-first neighbors. Concretely, we incorporate a differential (i.e., global minus depth-local) mechanism, which extracts depth-specific embeddings via another sequence model and subtracts them from global embeddings, emphasizing global timing behavior rather than depth-local influences. **(iv) Hybrid Graph-Sequence Architecture** to harness the complementary strengths of graph-based and sequence-based paradigms in the local breadth of nodes and the global depth of timing paths, respectively. Specifically, we propose an architecture that integrates GNNs into our sequence modeling pipeline. Within each block, input data is first processed by GNN modules to capture localized structural dependencies, followed by sequence modeling to encode long-range signal propagation. This integration enables end-to-end learning across RTL designs of varying structures.

In summary, our work makes the following key contributions:

(1) We present RTL-Sequencer, the first to propose a sequence-based paradigm to RTL timing prediction. This paradigm enables scalable learning of long-range signal dependencies with linear computational complexity, offering explicit signal directionality and elimination of intra-layer restriction.
(2) We design four synergistic techniques customized to the sequence-based paradigm, namely sequence shuffling, bidirectional sequence modeling, differentiable sequence modeling, and a hybrid graph–sequence architecture.
(3) We conduct extensive experiments to demonstrate the superiority of RTL-Sequencer over state-of-the-art graph-based approaches. In particular, comprehensive ablation studies confirm the effectiveness of the proposed four techniques.

2 PROBLEM FORMULATION

Our problem formulation follows prior works [9, 32]. Specifically, a digital circuit described at the RTL is represented as a directed graph $G = (V, E)$, where V denotes the set of nodes corresponding to registers and combinational logic elements, and E denotes the set of nodes corresponding to registers and combinational logic elements. This graph abstraction, often referred to as a Boolean Operator Graph (BOG) [9, 32], encapsulates the structural and functional dependencies inherent in RTL designs.

The primary objective is to predict the arrival time at each endpoint register node $n_{\text{ep}} \in V_{\text{ep}}$, denoted as $\text{AT}_{\text{pred}}(n_{\text{ep}})$, directly from the RTL representation. Ground-truth arrival times $\text{AT}_{\text{label}}(n_{\text{ep}})$ are obtained from sign-off static timing analysis (STA) performed after place-and-route. The prediction task is thus formulated as a supervised regression problem, where the goal is to minimize the discrepancy between $\text{AT}_{\text{pred}}(n_{\text{ep}})$ and $\text{AT}_{\text{label}}(n_{\text{ep}})$ across all register nodes. Formally, the optimization objective is expressed as:

$$\min_{\theta} \sum_{n_{\text{ep}} \in V} ||\text{AT}_{\text{pred}}(n_{\text{ep}}; \theta) - \text{AT}_{\text{label}}(n_{\text{ep}})||^2 \tag{1}$$

where $V_{\text{ep}} \subseteq V$ denotes the subset of endpoint register nodes, and θ represents the learnable parameters of the prediction model.

3 PRELIMINARY: SEQUENCE MODELS

In this section, we provide an overview of the background and motivation underlying modern linear sequence models [7, 12, 24, 28, 40]. In a general sequence model, the representation of a node is updated based on its current state and accumulated historical information. Notably, this historical context is losslessly encoded by preceding nodes while maintaining linear computational complexity. Such a formulation contrasts with attention-based architectures, in which the representation of each node is explicitly conditioned on all other nodes within the sequence, thereby incurring quadratic complexity.

Specifically, let an ordered sequence of nodes be denoted by $S = (n_1, n_2, ..., n_i, ..., n_N)$. At each block, $x_i \in \mathbb{R}^k$ is the input feature vector for node n_i, $y_i \in \mathbb{R}^k$ be the corresponding output vector, and $h_{i-1} \in \mathbb{R}^k$ be the hidden state encoding the history from n_1 to n_{i-1}. The update formula for node representation is governed by:

$$h_i = \texttt{Sequencer}(S, h_{i-1}, x_i) \tag{2}$$

$$y_i = \texttt{Projection}(h_i) \tag{3}$$

Here, the `Sequencer` captures the dependency between the current node n_i and the history encapsulated in h_{i-1} without strict

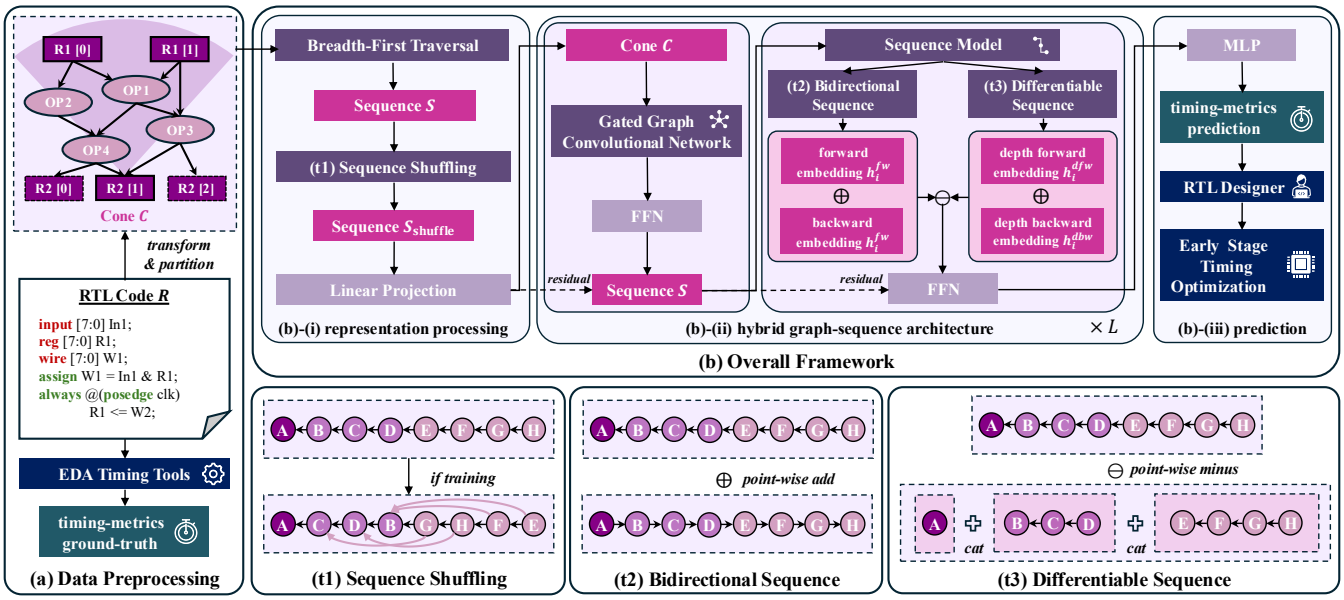


Figure 2: *Overview of RTL-Sequencer.* (a) RTL data preprocessing includes extracting ground-truth timing metrics via commercial EDA tools and transforming RTL codes into BOGs. Subsequently, logic cones are extracted from the BOG. (b) Overall architecture of the proposed RTL-Sequencer framework, where the extracted logic cone from (a) is fed into a sequence modeling pipeline to predict timing metrics and facilitate neural network training. (t1) Sequence Shuffling, (t2) Bidirectional Sequence, and (t3) Differentiable Sequence illustrate key components of our sequence-centric paradigm, each implemented as submodules within the architecture shown in (b), and designed to improve model generalization across heterogeneous RTL designs.

node-to-node restrictions. The `Projection` then refines channel-wise features of the updated hidden state h_i. The resulting y_i is regarded as the subsequent input for the following block, thereby facilitating representation learning across successive network blocks.

4 METHODOLOGY

In this section, we present the methodology of RTL-Sequencer and depict it in Fig. 2. To establish clarity and emphasize our contributions, we first outline a basic solution for our framework in Sec. 4.1, detailing how to transform RTL designs into node sequences and the subsequent application of sequence models for representation learning. Building on this foundation, we introduce four synergistic techniques, including sequence shuffling in Sec.4.2, bidirectional sequence modeling in Sec.4.3, differentiable sequence modeling in Sec.4.4, and a hybrid graph-sequence architecture in Sec.4.5, which further customize sequence models to RTL timing prediction.

4.1 Basic Solution

To establish the sequence-based paradigm, we first outline a basic solution of RTL-Sequencer in this section. This involves transforming RTL designs into ordered node sequences and applying sequence models to predict timing behavior directly from these sequences.

Initially, given that the timing behavior of an endpoint register is predominantly determined by its associated logic cone within the BOG, we adopt a cone-based partitioning strategy to isolate the cone, which is defined as the transitive fan-in subgraph rooted at the endpoint register. Formally, for each endpoint register ep $\in G$, we extract its logic cone $C \subseteq G$ via a backward traversal from ep to any input register, thereby capturing all upstream combinational and sequential elements that contribute to its signal arrival time.

Subsequently, we adopt an inverse Breadth-First Traversal (BFT) initiated from the endpoint to convert the cone C into the node sequence $S = \text{BFT}(C) = (n_1, n_2, ..., n_i, ..., n_N)$. In contrast to depth-first traversal, which may obscure timing dependencies due to its recursive exploration, BFT visits nodes in a level-wise order. This ordering preserves the hierarchical progression of signal propagation through the combinational gate and facilitates a more faithful representation of timing behaviour in the logic cone.

Such a node sequence S is input to the proposed sequence models, denoted as `Sequencer`, in place of conventional graph-based approaches. For each node $n_i \in S$, the forward hidden state embedding h_i^{fw} is computed by the sequence model, thereby capturing historical dependencies from preceding nodes in the sequence. Formally, the forward sequence embedding h_i^{fw} of n_i is defined as:

$$h_i^{\text{fw}} = \texttt{Sequencer}(S, h_{i-1}, x_i) \tag{4}$$

where h_{i-1} denotes the hidden state propagated from the earlier nodes and x_i represents the input feature vector associated with node n_i. In contrast to the basic formulation in Formula 2, the final representation h_i is not determined solely by $\texttt{Sequencer}(S, h_{i-1}, x_i)$; rather, it is augmented in subsequent stages through the integration of additional components that enrich the representational capacity.

In this section, we have introduced a basic solution for the RTL-Sequencer as a baseline. Nevertheless, the direct application of deterministic BFT and conventional sequence models proves suboptimal, as it often discards critical topological information when topology-invariant cones are reduced to fixed, unidirectional and breadth-first sequences. To overcome these limitations, the subsequent sections present four synergistic techniques that are designed

Feature	Type	Channel
cell type	one-hot	12
fan-out number	int	1
fan-in number	int	1
load capacitance	float	1
slew	float	1
node order in current BFT-level	int	1
number of nodes in current BFT-level	int	1
start flag of each BFT-level	one-hot	1
end flag of each BFT-level	one-hot	1
order of parent node in next BFT-level	int	1

Table 2: Overview of node representations as network inputs.

to further tailor sequence models to the challenging RTL timing prediction.

4.2 Sequence Shuffling

To address the information loss incurred when topology-invariant cones are converted into fixed node sequences, we introduce a stochastic shuffling mechanism. Specifically, each node ordering is randomized within each BFT depth level with 50% probability, while preserving the cone topology's invariance. Hierarchical consistency is maintained by synchronizing permutations across child nodes in subsequent levels. This procedure serves as a form of data augmentation, exposing sequence models to diverse node orderings while retaining logical dependencies and timing closure, ultimately enhancing generalization across heterogeneous RTL designs.

For example, consider a cone sequence $S = (x_1, x_2, x_3, x_4, x_5)$, where x_1 connects to $\{x_2, x_3\}$, and x_2, x_3 further connect to $\{x_4\}$, $\{x_5\}$, respectively. $S_{\text{shuffle}} = (x_1, x_3, x_2, x_5, x_4)$ can be a possible shuffled variant, in which the position of v_5 is elevated relative to v_4, reflecting the direct connection between x_3 and x_5.

During inference, it is necessary to determine a sorting strategy consistent with the training procedure. To fully leverage the information flow inherent in sequence models, we employ a heuristic deterministic ordering that ranks nodes within each breadth-first traversal (BFT) level according to their in-degree and out-degree statistics. Nodes exhibiting higher connectivity are positioned closer to the output register within each level, thereby enriching the contextual information available for timing prediction. This approach effectively integrates the advantages of stochastic data augmentation during training with the maximization of information utilization during inference, ultimately enhancing prediction accuracy across previously unseen RTL designs.

4.3 Bidirectional Sequence Modeling

In the baseline model described in Sec. 4.1, a single sequence model is employed to capture the forward sequence embedding h_i^{fw} for node n_i. This formulation, however, constrains information propagation to a unidirectional flow, thereby preventing later nodes from influencing earlier representations and limiting the depth of sequencer blocks. Such a restriction is also misaligned with practical design processes, as both Static Timing Analysis (STA) and logical synthesis inherently operate not in unidirectional contexts.

To overcome this limitation, we propose a bidirectional sequence modeling to augment the forward sequence S by its reversed counterpart, denoted as $S_{\text{flip}} = (n_N, ..., n_i, ..., n_2, n_1)$. A reversed sequence model is instantiated to process S_{flip}, thereby facilitating backward information flow. For each node n_i, the backward hidden state h_i^{bw} is computed by aggregating the outputs of backward sequencers:

$$h_i^{\text{bw}} = \texttt{Sequencer}(S_{\text{flip}}, h_{i+1}, x_i) \tag{5}$$

This dual-path formulation markedly enhances representational expressiveness by embedding each node within both its antecedent and subsequent nodes, thereby enabling the model to more effectively capture timing-critical dependencies across deep logics.

4.4 Differentiable Sequence Modeling

While breadth-first traversal avoids backtracking compared to depth-first traversal, it frequently places unrelated nodes at the same BFT depth, thereby placing them adjacently within the resulting node sequence. In sequence models, this adjacency causes each node's representation to be inevitably influenced by its immediate depth-level neighbors. As a result, sequence models tend to overemphasize local depth-specific structures while neglecting long-range dependencies along critical timing paths.

To address this limitation, we propose a differentiable mechanism that adaptively reduces irrelevant depth-specific context while accentuating global timing behaviors. Our design is inspired by recent advances in differentiable sequence modeling that employ embedding subtraction across entire sequences [41]. Distinct from these approaches, we introduce a sequence model tailored to operate on depth-specific subsequences rather than the full sequence. By subtracting depth-specific node embeddings from those obtained via the global sequence model, we derive depth-irrelevant embeddings that more faithfully preserve global timing dependencies.

For instance, consider the sequence $S = (n_1, n_2, n_3, n_4, n_5)$, where n_1 has children $\{n_2, n_3\}$, and n_2, n_3 connect to $\{n_4\}$, $\{n_5\}$, respectively. The corresponding BFT-level sequences D_i are defined as $D_1 = (N_1)$, $D_2 = D_3 = (n_2, n_3)$, $D_4 = D_5 = (n_4, n_5)$. For each node n_i, we first compute the depth-specific embeddings h_i^{dfw} and h_i^{dbw} from the forward and backward sequences as follows:

$$h_i^{\text{dfw}} = \texttt{Sequencer}(D_i, h_{i-1}, x_i) \tag{6}$$

$$h_i^{\text{dbw}} = \texttt{Sequencer}(D_{i\_\text{flip}}, h_{i+1}, x_i) \tag{7}$$

We further incorporate a self-gating mechanism to adaptively regulate the strength of the differential operation. Specifically, we introduce the learnable coefficient λ that quantifies the influence of depth-specific context. Owing to the `Sigmoid` activation function, λ is constrained to the interval $[0, 1]$. Ideally, $\lambda \approx 0$ indicates that the node embedding predominantly attends to global timing behavior, whereas $\lambda \approx 1$ reflects a stronger emphasis on depth-local structural information, allowing the representation to self-adjust through learning. The overall formulation is expressed as:

$$\lambda = \texttt{Sigmoid}(\texttt{MLP}(h_i^{\text{dfw}} + h_i^{\text{dbw}})) \tag{8}$$

$$h_i = h_i^{\text{fw}} + h_i^{\text{bw}} - \lambda(h_i^{\text{dfw}} + h_i^{\text{dbw}}) \tag{9}$$

Here, h_i^{fw} and h_i^{bw} denote the forward and backward global embeddings obtained from Sec. 4.1 and Sec.4.2, respectively. This differentiable strategy (i.e., global minus depth-local) ensures that node representations remain aligned with the global timing behavior rather than adaptively aligning with depth-specific structure.

4.5 Hybrid Architecture

While sequence models are effective in capturing long-range and deep dependencies, they can fail to localize and broader structural information surrounding nodes within design logic. To overcome

Model	Type	Arrival Time (AT)			Worst Negative Slack (WNS)			Total Negative Slack (TNS)		
		R	R^2	MAPE	R	R^2	MAPE	R	R^2	MAPE
GCN [20]	GNN	0.81	0.66	27.38%	0.78	0.62	30.00%	0.68	0.47	40.52%
GAT [31]	GNN	0.83	0.69	25.23%	0.81	0.65	27.46%	0.69	0.48	39.11%
SG-Former [35]	GT	0.73	0.54	32.67%	0.70	0.51	34.71%	0.66	0.43	46.80%
RTL-Timer [9]	XGBoost	0.85	0.72	23.55%	0.83	0.69	24.09%	0.70	0.50	37.10%
RTLDistill [32]	GNN-Propagation	0.89	0.81	21.86%	0.87	0.74	23.11%	0.74	0.56	32.15%
NUA-Timer [33]	GNN-Propagation	0.90	0.81	21.16%	0.88	0.77	22.62%	0.76	0.59	29.45%
CircuitFusion [8]	GT	0.91	0.83	20.29%	0.89	0.80	21.67%	0.82	0.69	26.16%
TF-Predictor [3]	Transformer	0.70	0.50	35.74%	0.67	0.45	37.98%	0.63	0.40	49.31%
RTL-Sequencer	Sequencer	0.92	0.85	17.24%	0.91	0.83	17.66%	0.88	0.77	23.92%

Table 3: Comparison between classic graph-based solutions, SOTA in the community, and RTL-Sequencer on timing prediction.

Model	Arrival Time (AT)			Worst Negative Slack (WNS)			Total Negative Slack (TNS)		
	ΔR	ΔR^2	ΔMAPE	ΔR	ΔR^2	ΔMAPE	ΔR	ΔR^2	ΔMAPE
RTL-Sequencer	0.92	0.85	17.24%	0.91	0.83	17.66%	0.88	0.76	23.92%
Basic Solution	-0.12	-0.22	+12.23%	-0.13	-0.21	+12.92%	-0.19	-0.26	+16.39%
w/o Stochastic Shuffling	-0.09	-0.14	+8.23%	-0.11	-0.18	+9.26%	-0.16	-0.22	+11.69%
w/o Bidirectional Sequence Modeling	-0.04	-0.09	+5.01%	-0.04	-0.05	+5.73%	-0.11	-0.15	+7.20%
w/o Differentiable Sequence Modeling	-0.01	-0.02	+2.37%	-0.02	-0.04	+2.78%	-0.06	-0.08	+3.46%
w/o Hybrid Architecture	-0.02	-0.02	+3.22%	-0.03	-0.05	+3.93%	-0.08	-0.11	+5.18%

Table 4: Ablation study of components in RTL-Sequencer compared to the full model.

this limitation, we propose a hybrid graph–sequence architecture that integrates the complementary strengths of GNNs for local context aggregation with the capacity of sequence models to capture global and deep temporal dependencies.

As illustrated in Fig. 2-(b), the proposed pipeline begins by applying BFT combined with sequence shuffling to the logic cone, thereby generating the initial sequence representation. This feature channel of sequence is subsequently projected into a higher-dimensional space through a linear transformation. Following this representation processing, a GNN is employed to capture broader local structural dependencies surrounding each node in the cone. The cone is subsequently converted back to an ordered sequence and processed by our sequence models, which are augmented with bidirectional and differential sequence modeling, to capture the long-range and deep timing dependencies. To further enrich channel-wise feature representations, feed-forward networks (FFNs) are appended to both the GNN and sequence models, following the design principles of Transformer-based architectures [30]. Crucially, all graph-to-sequence and sequence-to-graph transitions are implemented as differentiable operations, thereby enabling seamless end-to-end optimization through gradient descent.

5 EXPERIMENTS

5.1 Configurations

Dataset. We conduct experiments on 21 open-source RTL designs following RTL-Timer [9], employing 5-fold cross-validation. To ensure generalization, training and evaluation sets are strictly partitioned by designs. The dataset encompasses diverse mainstream hardware description languages, with design sizes ranging from 6K to 510K gates. For dataset generation, we utilize Synopsys Design Compiler for logic synthesis and Cadence Innovus for physical implementation, both targeting the NanGate45nm process design kit. Static timing analysis is performed using Synopsys PrimeTime to extract ground-truth timing metrics.

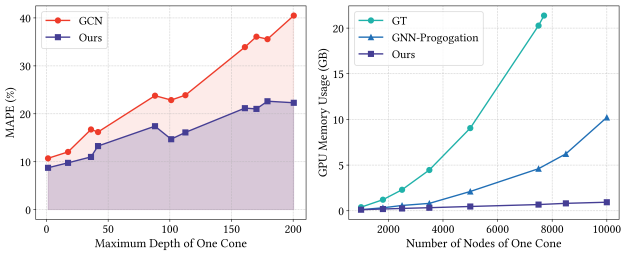
Evaluation Metrics. We evaluate model performance on three critical timing metrics: Arrival Time (AT), Worst Negative Slack (WNS), and Total Negative Slack (TNS). The following statistical measures are employed: **Pearson Correlation Coefficient (R)**: Quantifies the linear correlation between predicted and actual timing values. **Coefficient of Determination (R^2)**: Measures the proportion of variance in ground truth explained by the model. **Mean Absolute Percentage Error (MAPE)**: Captures the relative prediction error, with lower values indicating higher accuracy.

Implementation Details. We adopt the GatedGCN [21] and Mamba-2 [7] for our hybrid graph-sequence architecture. All models are trained on a cluster of 8× NVIDIA RTX 4090D using PyTorch Lightning, PyTorch Geometric, and Scikit-learn. We adopt the Adam optimizer with an initial learning rate of 1×10^{-3} and a per-GPU batch size of 8. For the model hyper-parameters, the channel number of the hidden state is 32, the number of sequencer blocks L is 3, and shuffling probability applied to each training sample is 50%.

Baseline. We benchmark our framework against three representative graph-based approaches, including GCN [20], GAT [31], and SG-Former [35]; as well as several SOTA methods widely adopted in the community, including RTL-Timer [9], RTLDistill [32], NUA-Timer [33], CircuitFusion [8], and TF-Predictor [3]. For consistency and fairness, all baselines are re-implemented within our setting, deliberately excluding auxiliary features derived from post-synthesis stages (e.g., layout information). This design choice highlights the contribution of architectural innovations to RTL timing prediction.

5.2 Results

As presented in Table 3, RTL-Sequencer consistently surpasses prior approaches across all evaluated timing metrics, thereby highlighting the efficacy of its architectural innovations. Specifically, the framework achieves the lowest AT MAPE of 17.24%, accompanied by the highest correlation scores, with $R = 0.92$ and $R^2 = 0.85$. In addition, RTL-Sequencer demonstrates marked superiority on TNS, attaining a MAPE of 23.92%, whereas the second-best method,



(a) MAPE Comparison of Arrival Time (b) Training GPU Memory Usage Comparison

Figure 3: Scalability comparison among GCN, GT, GNN-Propagation, and RTL-Sequencer demonstrates that our superior scalability in handling deeper cones with more nodes.

CircuitFusion [8], records 29.45%. This performance gap can be attributed to the absence of an explicit representation of signal propagation in attention mechanism of CircuitFusion, which only models node-to-node relationships. Although both CircuitFusion and SG-Former [35] adopt GT-based architectures, SG-Former yields a relatively poor AT MAPE of 32.67%, due to its reliance on a single block combining a GT with a GNN. RTLDistill [32] and NUA-Timer [33] are hindered by exponential accumulation of errors across cone layers, thus are outperformed by RTL-Sequencer. Notably, TF-Predictor [3] exhibits the weakest performance among all baselines, with an AT MAPE of 35.71%. Despite its claim as a sequence-based model, TF-Predictor relies on a naive Transformer architecture that focuses exclusively on specified timing paths. This design restricts its applicability to RTL stages and ultimately renders it ineffective for modeling meaningful representations of logic cones.

Moreover, Fig. 3 illustrates two fundamental limitations inherent in prior graph-based approaches. As shown in Fig. 3(a), the performance of GCN deteriorates more compared to RTL-Sequencer as logic depth increases, highlighting its restricted capacity to effectively capture deep combinational structures. In Fig. 3(b), GT exhibits quadratic memory overhead of the node count when the batch size is set to one, rendering it impractical for large-scale designs. Furthermore, by incrementally adding 100 nodes per layer within the cone, we demonstrate that the computational complexity of GNN-Propagation scales quadratically with cone depth. In contrast, RTL-Sequencer maintains linear memory complexity, thereby enabling scalable training across deep and complex designs.

5.3 Ablation Studies

To evaluate the individual contributions of the proposed architectural innovations, we conducted a comprehensive ablation study, the results of which are summarized in Table 4. The baseline solution can achieve accuracy comparable to graph-based approaches, thereby underscoring the promise of the sequence-based paradigm. Moreover, each component contributes measurable improvements in predictive performance, highlighting the synergistic nature of the overall framework. Among these, the sequence shuffling mechanism delivers the most pronounced gains in accuracy. This improvement arises from its function as a structural data augmentation strategy: by introducing stochastic permutations within BFT levels, the model is exposed to a wider range of topological variations while maintaining logical dependencies. Such diversity is able to mitigate overfitting to the deterministic node ordering, thereby enhancing generalization across heterogeneous RTL designs.

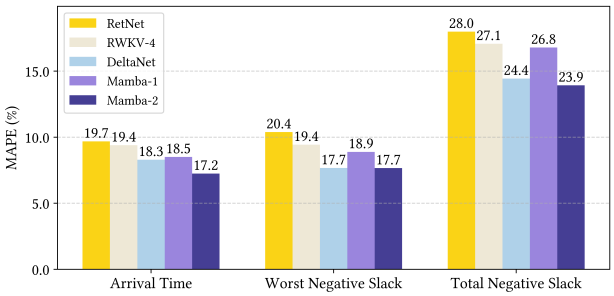


Figure 4: Comparison across sequence models. Mamba-2 achieves the lowest MAPE across all metrics. Notably, all sequence models surpass graph-based solutions.

To rigorously evaluate the effectiveness of the proposed sequence modeling paradigm for RTL timing prediction, we conducted a comprehensive comparative study across four representative sequence architectures. As illustrated in Fig. 4, the Mamba-2 model consistently delivers superior performance across all timing metrics, outperforming alternative sequence-based approaches. Notably, all sequence models substantially exceed traditional graph-based solutions, including GNN and GT variants, in both predictive accuracy and scalability. This consistent superiority highlights a fundamental paradigm shift from isotropic graph representations to directional sequence modeling, thereby substantiating the core hypothesis that logic cone linearization enables more expressive and scalable timing prediction. Collectively, these findings underscore the promise of sequence-driven frameworks as a robust and scalable alternative to graph-based methodologies for early-stage design automation.

6 CONCLUSION

This work introduces RTL-Sequencer, a novel sequence-based paradigm for scalable RTL timing prediction. By converting logic cones into ordered node sequences and employing advanced sequence modeling techniques, the framework achieves global receptive fields with linear computational complexity while explicitly preserving signal directionality. The integration of sequence shuffling, bidirectional sequence modeling, differentiable sequence modeling, and a hybrid graph–sequence architecture collectively enhances predictive accuracy and scalability, consistently outperforming SOTA graph-based approaches. Comprehensive experiments and ablation studies substantiate the effectiveness of both the overall paradigm and its individual components. Taken together, these results demonstrate the potential of RTL-Sequencer to advance early-stage timing estimation in digital design automation significantly.

7 ACKNOWLEDGEMENT

This work is supported by Hong Kong Research Grants Council (RGC) CRF-YCRG C6003-24Y, GRF 16200724, and T46-415/25-R. It 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.

REFERENCES

[1] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. 2019. Machine Learning-Based Pre-Routing Timing Prediction with Reduced Pessimism. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*.

[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001).

[3] Peng Cao, Guoqing He, and Tai Yang. 2022. Tf-predictor: Transformer-based prerouting path delay prediction framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 7 (2022).

[4] Peng Cao, Guoqing He, and Tai Yang. 2023. TF-Predictor: Transformer-Based Prerouting Path Delay Prediction Framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 7 (2023).

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*.

[6] Vidya A. Chhabria, Wenjing Jiang, Andrew B. Kahng, and Sachin S. Sapatnekar. 2023. A Machine Learning Approach to Improving Timing Consistency between Global Route and Detailed Route. *ACM Trans. Des. Autom. Electron. Syst.* 29, 1, Article 18 (Dec. 2023).

[7] Tri Dao and Albert Gu. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060* (2024).

[8] Wenji Fang, Shang Liu, Jing Wang, and Zhiyao Xie. 2025. Circuitfusion: multimodal circuit representation learning for agile chip design. *arXiv preprint arXiv:2505.02168* (2025).

[9] Wenji Fang, Shang Liu, Hongce Zhang, and Zhiyao Xie. 2024. Annotating slack directly on your verilog: Fine-grained rtl timing evaluation for early optimization. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*.

[10] Wenji Fang, Shang Liu, Hongce Zhang, and Zhiyao Xie. 2025. A Self-Supervised, Pre-Trained, and Cross-Stage-Aligned Circuit Encoder Provides a Foundation for Various Design Tasks. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASPDAC '25)*.

[11] Wenji Fang, Yao Lu, Shang Liu, Qijun Zhang, Ceyu Xu, Lisa Wu Wills, Hongce Zhang, and Zhiyao Xie. 2023. Masterrtl: A pre-synthesis ppa estimation framework for any rtl design. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*.

[12] Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*.

[13] Zizheng Guo, Mingjie Liu, Jiaqi Gu, Shuhan Zhang, David Z Pan, and Yibo Lin. 2022. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*.

[14] Zizheng Guo, Mingjie Liu, Jiaqi Gu, Shuhan Zhang, David Z. Pan, and Yibo Lin. 2022. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*.

[15] Xu He, Zhiyong Fu, Yao Wang, Chang Liu, and Yang Guo. 2022. Accurate timing prediction at placement stage with look-ahead RC network. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*.

[16] Tsung-Wei Huang and Martin DF Wong. 2015. OpenTimer: A high-performance timing analysis tool. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[17] Leilei Jin, Rongliang Fu, Zhen Zhuang, Liang Xiao, Fangzhou Liu, Bei Yu, and Tsung-Yi Ho. 2025. ChronoTE: Crosstalk-Aware Timing Estimation for Routing Optimization via Edge-Enhanced GNNs. In *2025 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[18] Andrew B. Kahng, Seokhyeong Kang, Hyein Lee, Siddhartha Nath, and Jyoti Wadhwani. 2013. Learning-based approximation of interconnect delay and slew in signoff timing tools. In *2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*.

[19] Andrew B. Kahng, Mulong Luo, and Siddhartha Nath. 2015. SI for free: machine learning of interconnect coupling delay and transition effects. In *2015 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*.

[20] TN Kipf. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[22] Rongjian Liang, Zhiyao Xie, Jinwook Jung, Vishnavi Chauha, Yiran Chen, Jiang Hu, Hua Xiang, and Gi-Joon Nam. 2020. Routing-Free Crosstalk Prediction. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[23] Daniela Sánchez Lopera and Wolfgang Ecker. 2022. Applying GNNs to Timing Estimation at RTL. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)*. Article 3.

[24] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048* (2023).

[25] Pei Quan, Yong Shi, Minglong Lei, Jiaxu Leng, Tianlin Zhang, and Lingfeng Niu. 2019. A Brief Review of Receptive Fields in Graph Convolutional Networks. In *IEEE/WIC/ACM International Conference on Web Intelligence - Companion Volume* (Thessaloniki, Greece) *(WI '19 Companion)*.

[26] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. 2023. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993* (2023).

[27] Prianka Sengupta, Aakash Tyagi, Yiran Chen, and Jiang Hu. 2022. How Good Is Your Verilog RTL Code? A Quick Answer from Machine Learning. In *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[28] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621* (2023).

[29] Veronika Thost and Jie Chen. 2021. Directed acyclic graph neural networks. *arXiv preprint arXiv:2101.07965* (2021).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[32] Mingjun Wang, Yihan Wen, Bin Sun, Jianan Mu, Juan Li, Xiaoyi Wang, Jing Justin Ye, Bei Yu, and Huawei Li. [n. d.]. Bridging Layout and RTL: Knowledge Distillation based Timing Prediction. In *Forty-second International Conference on Machine Learning*.

[33] Ziyi Wang, Fangzhou Liu, Tsung-Yi Ho, David Pan, and Bei Yu. 2025. NUA-Timer: Pre-Synthesis Timing Prediction Under Non-Uniform Input Arrival Times. In *2025 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[34] Ziyi Wang, Siting Liu, Yuan Pu, Song Chen, Tsung-Yi Ho, and Bei Yu. 2023. Restructure-Tolerant Timing Prediction via Multimodal Fusion. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*.

[35] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems* 36 (2023).

[36] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Chen-Chia Chang, Jingyu Pan, and Yiran Chen. 2022. Preplacement Net Length and Timing Estimation by Customized Graph Neural Network. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022).

[37] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Yixiao Duan, and Yiran Chen. 2021. Net2: A graph attention network method customized for pre-placement net length estimation. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*.

[38] Ceyu Xu, Chris Kjellqvist, and Lisa Wu Wills. 2022. SNS's not a synthesizer: a deep-learning-based synthesis predictor. In *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA '22)*.

[39] Ceyu Xu, Pragya Sharma, Tianshu Wang, and Lisa Wu Wills. 2023. Fast, Robust and Transferable Prediction for Hardware Logic Synthesis. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23)*.

[40] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. 2024. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems* 37 (2024).

[41] Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. 2024. Differential transformer. *arXiv preprint arXiv:2410.05258* (2024).

[42] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in neural information processing systems* 32 (2019).

[43] Xinyun Zhang, Binwu Zhu, Fangzhou Liu, Ziyi Wang, Peng Xu, Hong Xu, and Bei Yu. 2024. Disentangle, Align and Generalize: Learning A Timing Predictor from Different Technology Nodes. In *Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24)*. Article 133.

[44] Ruizhe Zhong, Junjie Ye, Zhentao Tang, Shixiong Kai, Mingxuan Yuan, Jianye Hao, and Junchi Yan. 2024. PreRoutGNN for timing prediction with order preserving partition: global circuit pre-training, local delay learning and attentional cell modeling. Article 1905.