Learning-Driven Physically Aware Large-Scale Circuit Gate Sizing

Yuyang Ye, Peng Xu, Lizheng Ren, Tinghuan Chen, *Member, IEEE*, Hao Yan, *Member, IEEE*, Bei Yu, *Senior Member, IEEE*, and Longxing Shi, *Senior Member, IEEE*

***Abstract*—Gate sizing plays an important role in timing optimization after physical design. Existing machine learning-based gate sizing works cannot optimize timing on multiple timing paths simultaneously and neglect the physical constraint on layouts. They cause suboptimal sizing solutions and low-efficiency issues when compared with commercial gate sizing tools. In this work, we propose a learning-driven physically aware gate sizing framework to optimize timing performance on large-scale circuits efficiently. In our gradient descent optimization-based work, for obtaining accurate gradients, a multimodal gate sizing-aware timing model is achieved via learning timing information on multiple timing paths and physical information on multiple-scaled layouts jointly. Then, gradient generation based on the sizing-oriented estimator and adaptive back-propagation are developed to update gate sizes. Our results demonstrate that our work achieves higher-timing performance improvements in a faster way compared with the commercial gate sizing tool.**

***Index Terms*—Deep learning, design automation, design for quality.**

I. Introduction

GATE sizing on post-routing circuits is fundamental for timing optimization to achieve sign-off timing closure with smaller the worst-negative slack (WNS) and total negative slack (TNS). The solution space scales exponentially with respect to the size of circuits [1], [2]. Under advanced technology, as illustrated in Fig. 1(a), physically aware timing engineering change order (ECO) flow is proposed. The flow can consider physical information and timing information jointly to achieve timing closure [3]. However, poor convergence forces engineers to perform many time-consuming iterations throughout the flow [4]. It makes an efficiency bottleneck for gate sizing.

Existing gate sizing algorithms can be divided into two kinds.

1) *Analytical Methods [2], [5], [6], [7], [8], [9], [10]:* Discrete gate sizing is solved through gradient descent optimization using Lagrangian relaxation-based algorithms in these methods.
2) *Machine-Learning Methods [4], [11], [12]:* Machine learning models are used to perform gate sizing through modeling circuits. Although machine learning has achieved many improvements in the gate-sizing problem, the performance of previous works cannot meet industry requirements when compared with commercial EDA tools. In RL-sizer [11], the generalization ability and runtime costs limit the application. In TranSizer [4], optimization performance is sensitive to the accuracy of the proposed gate sizing prediction model. A small prediction error that happens on critical paths always causes terrible optimization results, which makes TranSizer difficult to achieve stable and really optimal performance.

Recently, learning-driven gradient descent optimization works have solved some EDA issues [13], [14], [15], [16], [17], [18]. Fortunately, it is also a good idea for gate-sizing which combines analytical and machine learning methods jointly. However, totally different from other EDA problems, it is a special task to achieve gate-sizing based on learning-driven gradient descent optimization. There are two main challenges.

1) Achieving a gate sizing-aware timing model where accurate optimization gradients are calculated based on it.
2) Generating and back-propagating gradients w.r.t. discrete gate sizes on large-scale circuits efficiently and effectively.

For challenge (1), modeling gate sizing-induced timing performance variations urgently needs timing information on paths and physical information on layouts. For timing paths, the path delay variations of multiple timing paths caused by gate sizing on one single gate always are different [2]. As shown in Fig. 1(b), Path 1 and Path 2 are critical paths that go through gate U4. The setup timing performance of Path 1 can be optimized through upsizing gate U4. However, the larger effective capacitance of up-sized U4 loaded on gate U2 and U1 causes delay degradation on Path 2. The tradeoff between the optimization and degradation on multiple critical paths should

Received 11 March 2024; revised 20 July 2024 and 21 October 2024; accepted 27 October 2024. Date of publication 30 October 2024; date of current version 23 April 2025. This work was supported in part by the National Key R&D Program of China under Grant 2023YFB4402900; in part by the National Natural Science Foundation of China under Grant 62274034, Grant 62304197, and Grant 62474038; in part by the Key Research and Development Program of Jiangsu Province under Grant BE2023003-2; and in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14208021. This article was recommended by Associate Editor I. H.-R. Jiang. *(Corresponding author: Hao Yan.)*

Yuyang Ye, Peng Xu, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR.

Lizheng Ren, Hao Yan, and Longxing Shi are with the National ASIC Research Center, Southeast University, Nanjing 210096, China, and also with National Center of Technology Innovation for EDA, Nanjing, China (e-mail: yanhao@seu.edu.cn).

Tinghuan Chen is with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China.

Digital Object Identifier 10.1109/TCAD.2024.3488577

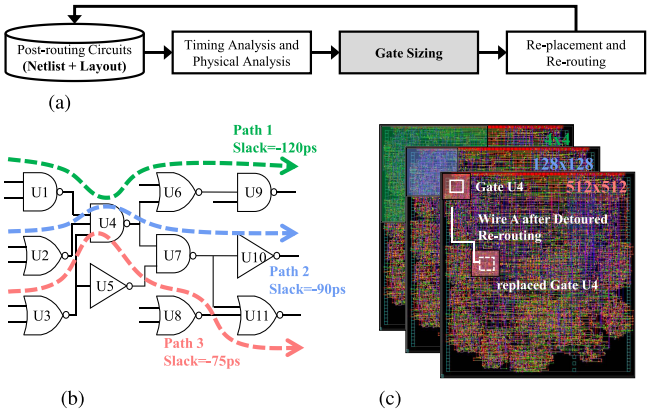

Fig. 1. Rich information 1) in optimization flow; 2) on timing paths; and 3) on design layouts. (a) Classical gate sizing flow where many iterations are necessities. (b) Netlist with multiple paths. (c) Layout under multiple scales.

be achieved while gate sizing. On design layouts, gate sizing might cause wire delay degradations after replacement and rerouting. As shown in Fig. 1(c), when replacement happens on gate U4 in the region with high-gate density, up-sized gate U4 must be replaced to avoid overlapping on layouts. In the rerouting stage, on the layout with high-wire congestion, the wire length of wire A increases due to detoured routing. After that, the delay of wire A degrades with the wire lengths. For layouts under multiple scales, the results of replacement and rerouting are different [1]. Thus, physical information on multiple-scaled layouts is important to tradeoff wire delay degradations. In addition, different from prediction works, the target of the timing model used in our work is to achieve optimal gate-sizing. The optimization information from the commercial gate sizing tool can be considered to guide the gradient. In summary, timing information on paths, physical information on layouts and optimization information in the industrial flow should be given full and joint consideration.

For challenge (2), the size of each gate is discrete rather than continuous. It means round functions should be used in timing models based on achievable gate sizes. However, round functions are not differentiable. AGD [15] proposed to use the Softmax functions to replace round functions for approximating the gradients in discrete functions as categorical variables. However, the gate size should be regarded as an integer-valued variable to retain the relationships between different sizes. Inspired by recent quantization-aware training works, the straight-through estimator can help us to obtain accurate gradients w.r.t. discrete gate sizes [19], [20], [21]. It is helpful to avoid discrepancies between the forward and backward pass, leading to global optimal gate sizing results. In addition, on large-scale circuits with numerous gates, there is a high-dimensional issue during gradient back-propagation. When numerous gates update sizes simultaneously, there are interdependencies among them. Limited considerations about the problem cause low-optimization efficacy.

In this work, we propose a learning-driven physically aware gate sizing framework to achieve timing optimization on large-scale circuits efficiently. Our work overcomes the above challenges of achieving gate sizing via gradient descent optimization. To obtain a gate sizing-aware timing model, we learn optimization information, timing information on paths and physical information on layouts jointly through multimodal learning. The learned information helps to accurately model timing optimization and degradation induced by gate sizing. To update gate sizes based on gradients effectively, we generate accurate timing performance gradients w.r.t. integer-valued gate sizes and back-propagate them with different priorities on different gates. We highlight our contributions.

1) For the first time, we propose a learning-driven framework to achieve physically aware gate-sizing. It can optimize timing performance effectively on large-scale circuits.
2) We achieve multimodal gate sizing-aware timing modeling via timing information aggregation on multiple critical paths and physical information aggregation on multiple scaled layouts. The optimization information from Synopsys IC-Compiler II (*ICC*2) [3] is utilized in training to guide the gradients of our timing model.
3) We perform gate sizing based on the size gradients of our timing model. A sizing-oriented straight-through estimator is developed to efficiently generate size gradients in discrete functions. An adaptive gradient back-propagation method is presented to update gate sizes effectively.
4) Our framework is evaluated with open-source designs in TSMC 16-nm technology. The results demonstrate that it can achieve 16.29%/18.61% TNS/WNS improvements and 6.64× speedup on average compared with the commercial gate sizing tool *ICC*2.

II. Preliminaries

A. Timing Optimization

Timing optimization is important in the circuit design flow to fix timing issues on timing paths. In circuits, timing paths are composed of a startpoint and an endpoint. The startpoint is a primary input or a register's output pin, while the endpoint is a primary output or a register's input pin. And the path slacks of all paths are computed based on path delays and the target clock period. Two metrics is used to evaluate timing performance, including 1) the TNS, which is the sum of the negative slacks observed at the primary outputs of the circuit and 2) the WNS, which is the WNS observed among all primary outputs of the circuit. Timing optimization focus on improving timing performance through bringing changes to circuits. In our work, we look forward to achieving it through gate sizing. It is a representative technique [1]. It chooses a better size for each gate from the cell library to optimize overall timing performance. Modern physically aware gate sizing flows should not only consider timing information but also physical information, e.g., gate density and wire congestion, to avoid numerous iterations.

B. Important Definitions

We give some important definitions as follows.

Definition 1 (Gatewise Critical Path): The most critical timing path through the target gate.

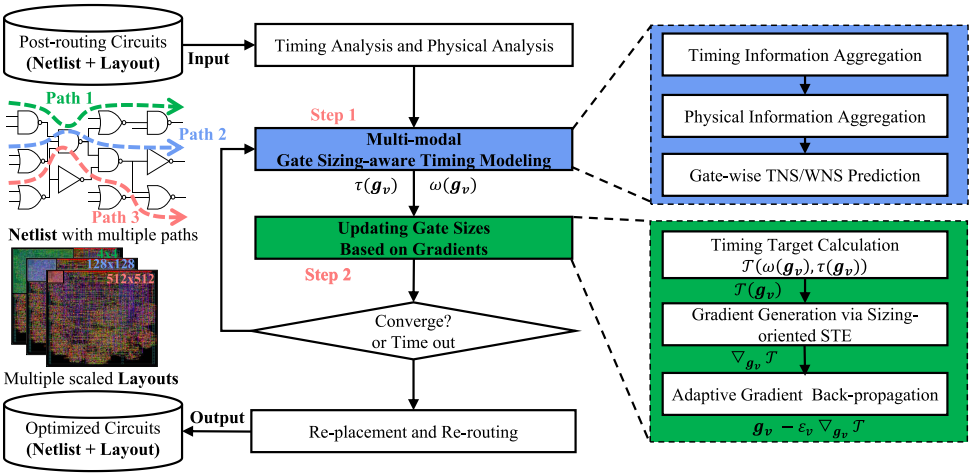

Fig. 2. Overall flow of our framework.

Definition 2 (Gatewise Path Group): The path group that is composed of critical timing paths through the target gate.

Definition 3 (Gatewise WNS): The negative slack of gatewise critical path for the target gate.

Definition 4 (Gatewise TNS): The TNS of paths in the gatewise path group for the target gate.

Examples: As shown in Fig. 1(b), Path 1, Path 2 and Path 3 are the gatewise critical paths of gate U4, U7 and U3, respectively. For the gate U4, Path 2 and Path 3 are included in the gatewise path group for it. The gatewise WNS of gate U4 equals to -120 ps. The gatewise TNS of gate U4 equals to -285 ps.

C. Problem Formulation

Based on these definitions, the problem of gate sizing can be formulated as follows.

Problem 1 (Gate Sizing): Given a post-routing netlist with timing information on multiple critical timing paths and layout with physical information under multiple scales, our target is to achieve optimal gate sizes of all gates $\{\boldsymbol{g}_v, v \in \mathcal{V}\}$ based on the information to obtain optimized timing performance with smaller TNS and WNS, where $\mathcal{V}$ is the gate set.

III. Overall Flow

As illustrated in Fig. 2, we first briefly introduce the overall flow of our gate sizing framework. The proposed framework can be divided into two steps: step 1 achieves the gate sizing-aware timing modeling based on multimodal learning (Section IV) and step 2 updates gate size based on gradients (Section V). In step 1, we learn timing information on multiple paths through timing feature aggregation and physical information on multiple scaled layouts through physical feature aggregation jointly. Based on learned information, we perform gatewise TNS $\tau(\boldsymbol{g}_v)$ and WNS $\omega(\boldsymbol{g}_v)$ prediction where slack labels and gradient labels are used in the loss function to ensure high accuracy. In step 2, we calculate the timing target $\mathcal{T}(\omega(\boldsymbol{g}_v), \tau(\boldsymbol{g}_v))$ based on our timing model. Then we generate timing target gradients $\nabla_{\boldsymbol{g}_v}\mathcal{T}$ w.r.t., gate sizes $\{\boldsymbol{g}_v, v \in \mathcal{V}\}$. The sizing-oriented straight-through estimator helps to solve discrete issues. Finally, we update the gate size of each gate $(\boldsymbol{g}_v - \varepsilon_v \nabla_{\boldsymbol{g}_v}\mathcal{T})$ via the adaptive gradient backward propagation. Our framework can optimize the timing performance of circuits, including TNS and WNS. The details are discussed as follows.

IV. Gate Sizing-Aware Timing Modeling

A. Date Representation

Timing Features on Netlists: As shown in Fig. 3(a), we transfer the circuit netlist to a graph $\mathbb{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$ consisting of a node set $(\mathcal{V})$, a edge set $(\mathcal{E})$ and a subgraph set $(\mathcal{P})$. Nodes are gates and edges are wires. More importantly, subgraphs are critical paths composed of gates and wires on paths. The circuit graph $\mathbb{G}$ is represented with node feature matrix $\boldsymbol{X}^T$: $\{\boldsymbol{x}_v^T, v \in \mathcal{V}\}$, adjacency matrix $\boldsymbol{J}$. The details of features in feature vector $\boldsymbol{x}_v^T$ includes the following.

1) Current gate size $\{\boldsymbol{g}_v, v \in \mathcal{V}\}$: Extracted by the cell type name, determining the driving strength of the gate.
2) *Gate Type:* For example, NAND, NOR, and embedded as a one-hot vector.
3) *Wire Capacitance and Resistance:* Extracted from the SPEF files generated by StarRC [22].
4) *Pin Capacitance:* Extracted from the timing library.
5) *Original Gate Size:* Extracted from the cell type name of circuits before gate sizing.

Physical Features on Layouts: We divide the overall layout into different scales with $M \times N$ grid cells. In previous timing models [23], they work on layout under one scale. Thus, the values of M and N are set to be constant, which equals 512. Achieving gate sizing based on physical information on different scaled layouts can obtain different results [3]. Thus, we collect physical features on multiple-scaled layouts where M and N are set to be different values. Specifically, local and global physical information is collected on large-scale and small-scale layouts, respectively. The detailed considered physical features should be closely correlated with gate sizing, which include: 1) vertical wire congestion; 2) horizontal wire congestion; and 3) gate density. Fig. 3(b) gives examples of

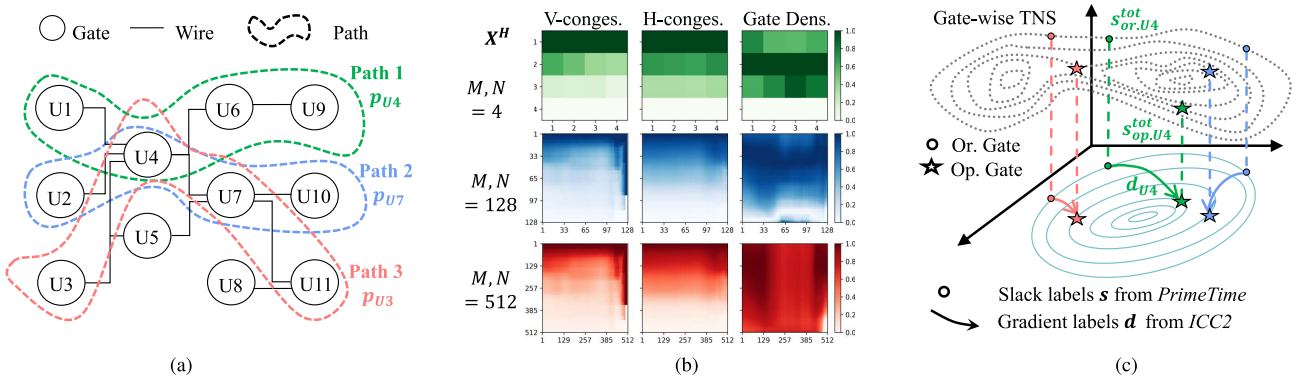

Fig. 3. Data representation in our work. "Or. Gate" and "Op. Gate" represent the original gate size and optimized gate size. (a) Circuit netlist graph. (b) Physical features of multiple scaled layouts. (c) Gatewise TNS labels and TNS gradient labels.

physical features $\boldsymbol{X}^H$: $\{\boldsymbol{x}_v^H, v \in \mathcal{V}\}$ on different scaled layouts for one Opencore design NOVA.

Slack Labels and Gradient Labels: As shown in Fig. 3(c), there are two kinds of labels used for training our timing model, including slack labels and gradient labels. Slack labels $\{s_v^{\text{tot}}, s_v^{wst}; v \in \mathcal{V}\}$ are gatewise total slacks and gatewise WNSs for all gates, which are generated via static timing analysis based on Synopsys PrimeTime [24]; Different from previous works, the gradient labels $\{d_v^{\text{tot}}$ and $d_v^{wst}, v \in \mathcal{V}\}$ are the gate sizing directions to optimize TNS and WNS, which are generated based on gate sizing results of *ICC*2. For gate v, given the original and *ICC*2 optimized gate sizes $\{\boldsymbol{g}_{or.v}, \boldsymbol{g}_{op.v}\}$ and gatewise TNS results $\{s_{or.v}^{\text{tot}}, s_{op.v}^{\text{tot}}\}$, d_v^{tot} can be computed as: $d_v^{\text{tot}} = (s_{op.v}^{\text{tot}} - s_{or.v}^{\text{tot}})/(\boldsymbol{g}_{op.v} - \boldsymbol{g}_{or.v})$. In the same way, d_v^{wst} is obtained. Our work focuses on achieving timing optimization by gate sizing. The optimization direction of *ICC*2 is the best possible after many explorations. Thus, the gradient labels can help speed up the optimization process and avoid local optimal problems.

B. Timing Feature Aggregation

Gate sizing for one target gate should consider all critical paths through it to achieve timing optimization and degradation tradeoff. One example of the timing information aggregation flow for gate U4, gate U7 and gate U3 is shown in Fig. 4. Given original timing features $\boldsymbol{X}^T$: $\{\boldsymbol{x}_v^T, v \in \mathcal{V}\}$ as input, the flow outputs path aggregated timing features $\boldsymbol{T}$: $\{\boldsymbol{t}_v, v \in \mathcal{V}\}$. For one gate v, the gatewise critical path and path group of gate v are p_v and $\mathcal{P}_v$, respectively. The detailed progress in generating $\boldsymbol{t}_v$ is discussed as follows.

Timing Feature Encoder: Given the input timing features $\{\boldsymbol{x}_v^T, v \in \mathcal{V}\}$, Transformers achieve timing feature encoding path by path and output $\{\boldsymbol{t}_v^p, v \in \mathcal{V}, p \in \mathcal{P}\}$. On path p, the $\boldsymbol{t}_v^p$ is generated via

$$\boldsymbol{t}_v^p = \text{Transformer}\left(\left\{\boldsymbol{x}_u^T, u \in \mathcal{N}_p\right\}, \boldsymbol{x}_v^T\right) \tag{1}$$

where $\mathcal{N}_p$ is the gate set of path p. On gatewise critical path p_v, the results of timing feature encoding $t_v^{p_v}$ is regarded as the critical encoding of gate v. Here, Transformer proposed in TransSizer [4] is used in our work. This part can collect timing information in a path-by-path way.

Path-Based Timing Feature Fusion: In the timing feature encoder, similar to TransSizer [4], the timing feature on paths is learned path by path. However, for real optimal gate sizing, timing performances on all critical timing paths through one gate should be considered jointly to achieve timing optimization and degradation tradeoff. In this work, we focus on achieving timing information aggregation on multiple timing paths. The final path aggregated timing feature $\boldsymbol{t}_v$ of gate v is composed of three parts, including critical encoding, intrapath encoding, and interpath encoding.

1) In the critical encoding part, we obtain the timing feature encoding result $\boldsymbol{t}_v^{p_v}$ of gate v on its gatewise critical path p_v. The slack of p_v is gatewise WNS of v and is dominant in gatewise TNS. The influence of gate-sizing happened on gate v on path p_v is modeled accurately in this part. Thus, it helps improve the prediction accuracy of gatewise WNS and TNS efficiently.
2) In the intrapath encoding part, we obtain the timing feature encoding results of gatewise critical path p_v via pooling all gates' timing feature encoding results on it. It helps our timing model to capture the relationship between gate v and other gates on p_v for accurate gatewise WNS and TNS predictions.
3) In the interpath encoding part, we combine the timing feature encoding results of gate v on all critical paths in gatewise path group $\mathcal{P}_v$ through average pooling. This part captures the relationships between gate v and all critical paths through it. It achieves accurately modeling the timing variations on paths in $\mathcal{P}_v$ caused by gate sizing on gate v.

It is helpful to achieve accurate gatewise TNS prediction. The path aggregated timing feature $\boldsymbol{t}_v$ is computed as

$$\boldsymbol{t}_v = \underbrace{\left(\boldsymbol{t}_v^{p_v}, p_v \text{ is critical path}\right)}_{\text{Critical Encoding}} \left\| \underbrace{\text{SUM}\left(t_{u_v}^{p_v}, u_v \in \mathcal{N}_{p_v}\right)}_{\text{Intra-path Encoding}} \right\| \underbrace{\text{AVE}\left(t_v^p, p \in \mathcal{P}_v\right)}_{\text{Interpath Encoding}} \tag{2}$$

where p_v is the gatewise critical path of gate v and $\mathcal{N}_{p_v}$ is the gate set of path p_v. $\mathcal{P}_v$ is the gatewise path group of gate v.

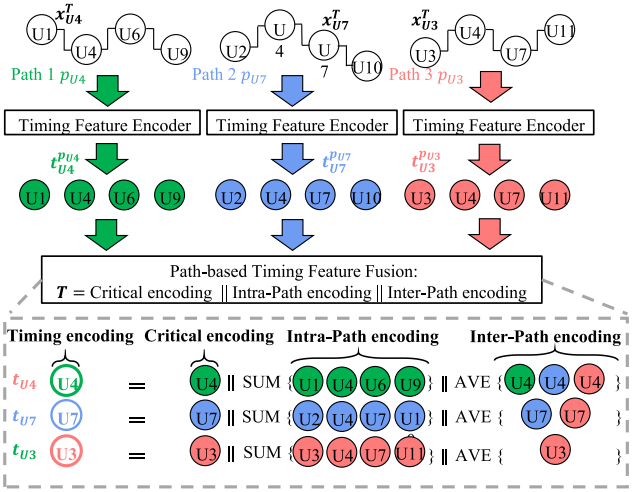

Fig. 4. Example of timing information aggregation on multiple paths.

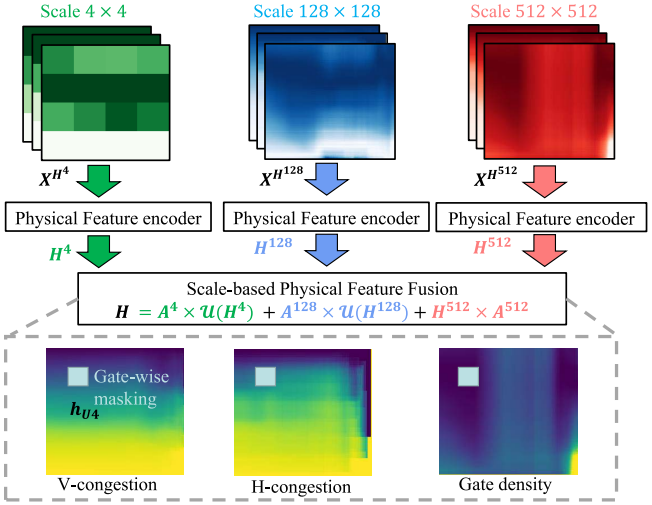

Fig. 5. Example of physical information aggregation on multiple scaled layouts.

SUM and AVE represent sum pooling and average pooling operations, respectively.

C. Physical Feature Aggregation

Aggregating the differentiated information on different scales benefits capturing the circuit timing variation caused by replacement and rerouting after gate sizing. One example of the physical information aggregation flow for layouts under 4×4, 128×128 and 512×512 scales is illustrated in the right part of Fig. 5. Given the input physical feature $\boldsymbol{X}^{H^{M\times N}}$, the flow outputs the scale aggregated physical feature $\boldsymbol{H}$ through combining information on different-scaled layouts. Since M equals to N in this example, $\boldsymbol{X}^{H^{M\times N}}$ and $\boldsymbol{H}^{M\times N}$ can be represented with $\boldsymbol{X}^{H^M}$ and $\boldsymbol{H}^M$. The flow is divided into two modules: 1) the physical feature encoder module and 2) the scale-based physical feature fusion module. The flow captures global and local physical information jointly on layouts.

Physical Feature Encoder: We start by encoding physical features under different scales independently and generate $\boldsymbol{H}^{M\times N}$. For the tradeoff between efficiency and effectiveness, the ResNet [25] and ASPP [26] are used to extract and compress input physical features, respectively. Specially, ResNet layer is constructed based on the feature extraction part of ResNet-50 without all other necessary parts. The ASPP layer is composed of five "Conv-BN-ReLU" branches. The kernel sizes and dilation rates of them are 1; 3; 3; 3; 1 and 1; 2; 5; 7; 1. All convolution operations use the padding to ensure that the input and output sizes are consistent. A global average pooling operation and an up-sampling operation are used before and after the second branch to capture the global physical information and restore it to the original size. All results of the five branches are concatenated along the channel dimension and fused by a branch to obtain the output. Thus, $\boldsymbol{H}^{M\times N}$ can be computed as

$$\boldsymbol{H}^{M\times N} = \text{ResNet-ASPP}\left(\boldsymbol{X}^{H^{M\times N}}\right). \tag{3}$$

Next, these features are fed successively to the scale-based physical feature fusion module for subsequent processing.

Scale-Based Physical Feature Fusion: We set one main scale, which equals 512×512, the biggest scale we selected in our work. For physical features on the small-scaled layout, we directly up-sample them by the bi-linear interpolation. Based on all encoded physical features from multiple scaled layouts $\{\boldsymbol{H}^{1\times 1}, \boldsymbol{H}^{2\times 2}, \ldots, \boldsymbol{H}^{256\times 256}, \boldsymbol{H}^{512\times 512}\}$, the scale attention $\boldsymbol{A}$ corresponding to each scale can be obtained. The process is formulated as

$$\boldsymbol{A} = \sigma\left(\Psi\left\{\mathcal{U}\left(\boldsymbol{H}^{1\times 1}\right)||\ldots||\boldsymbol{H}^{512\times 512}\right\}\right) \tag{4}$$

where Ψ indicates the stacked "Conv-BN-ReLU" layers which are commonly used in convolutional neural networks [25]. $||$ represents the concatenation operations. $\mathcal{U}(\cdot)$ refers to the bi-linear interpolation operations for up-sampling mentioned above. σ is Softmax activation operation in our work. Based on generated scale attention, we can obtain the final scale aggregated physical feature $\boldsymbol{H}$ by combining the scale-specific information jointly. Inspired by [23], gatewise masking can help us to get scale aggregated physical feature for each gate $\{\boldsymbol{h}_v, v \in \mathcal{V}\}$. They can be computed as

$$\boldsymbol{H} = \sum_{\text{all scales}} \boldsymbol{A}^{M\times N} \times \mathcal{U}\left(\boldsymbol{H}^{M\times N}\right), \quad \boldsymbol{h}_v = \boldsymbol{M}_v \boldsymbol{H} \tag{5}$$

where $\mathcal{U}(\cdot)$ is unnecessary for the features on main scale $\boldsymbol{H}^{512\times 512}$. $\boldsymbol{M}_v$ is the gatewise mask for gate v. These designs can selectively aggregate the scale-specific physical features to explore subtle but critical information among different scales. It helps predict TNS and WNS improvements and degradations induced by gate sizing after replacement and rerouting.

D. Gatewise TNS and WNS Prediction

Based on the path aggregated timing features $\boldsymbol{T}$: $\{\boldsymbol{t}_v, v \in \mathcal{V}\}$ and scale aggregated physical features $\boldsymbol{H}$: $\{\boldsymbol{h}_v, v \in \mathcal{V}\}$, we use multilayer perceptron layers MLP^{τ} and MLP^{ω} to predict the gatewise TNSs and gatewise WNSs for all gates

$$\tau\left(\boldsymbol{g}_v\right) = MLP^{\tau}\left(\boldsymbol{t}_v||\boldsymbol{h}_v\right), \quad \omega\left(\boldsymbol{g}_v\right) = MLP^{\omega}\left(\boldsymbol{t}_v||\boldsymbol{h}_v\right) \tag{6}$$

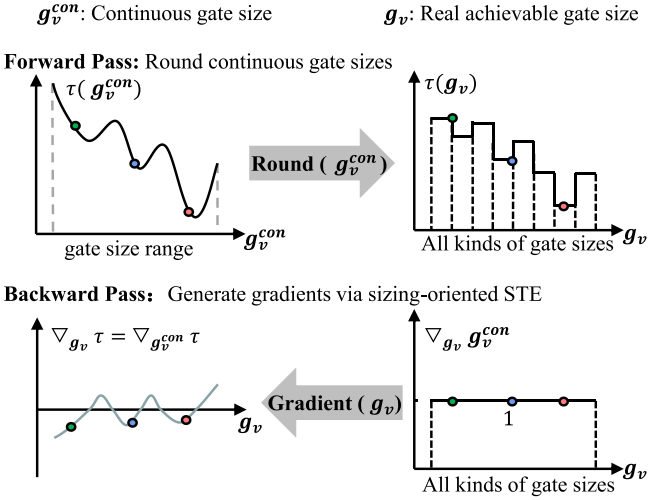

Fig. 6. Example flow of generating gradients of gatewise TNS w.r.t. gate sizes via the sizing-oriented straight-through estimator (STE).

where $\boldsymbol{g}_v$ is the gate size of gate v. Both the slack labels and gradient labels are used in loss functions of $\tau(\boldsymbol{g}_v)$ and $\omega(\boldsymbol{g}_v)$. The slack labels play important and fundamental roles in improving timing model accuracy. The gradient labels can be regarded as constraints to guide optimization directions. Combining these two labels, the loss functions used for training are illustrated in

$$\mathcal{L}^{\tau} = \sum_{v\in\mathcal{V}} \left\{ \underbrace{\left(s_v^{\text{tot}} - \tau(\boldsymbol{g}_v)\right)^2}_{\text{slack labels}} + \underbrace{\left(d_v^{\text{tot}} - \nabla_{\boldsymbol{g}_v}\tau\right)^2}_{\text{gradient labels}} \right\}$$
$$\mathcal{L}^{\omega} = \sum_{v\in\mathcal{V}} \left\{ \underbrace{\left(s_v^{wst} - \omega(\boldsymbol{g}_v)\right)^2}_{\text{slack labels}} + \underbrace{\left(d_v^{wst} - \nabla_{\boldsymbol{g}_v}\omega\right)^2}_{\text{gradient labels}} \right\} \quad (7)$$

where s_v^{tot} and s_v^{wst} are gatewise total slacks and gatewise WNSs generated via Synopsys PrimeTime; d_v^{tot} and d_v^{wst} are gradients of gatewise total slacks and gatewise WNSs generated after Synopsys *ICC2* gate sizing; and $\nabla_{\boldsymbol{g}_v}\tau$ and $\nabla_{\boldsymbol{g}_v}\omega$ are gradients of $\tau(\boldsymbol{g}_v)$ and $\omega(\boldsymbol{g}_v)$ w.r.t. gate size, where the detailed flow to generate them is discussed in Section V-B.

V. Updating Gate Size Based on Gradients

A. Timing Target Calculation

After obtaining the well-trained $\tau(\boldsymbol{g}_v)$ and $\omega(\boldsymbol{g}_v)$, the timing target $\mathcal{T}(\boldsymbol{g}_v)$ for gate sizing is calculated based on predicted gatewise TNS and WNS results. Different from previous work [15], we consider all critical paths in the timing target $\mathcal{T}(\boldsymbol{g}_v)$ rather than the worst path on each point. This is because focusing on optimizing the worst path on each point might cause timing degradations on other critical paths on the same point. It makes many time-consuming iterations and causes local optimal problems in other works [4]. The timing target $\mathcal{T}(\boldsymbol{g}_v)$ can computed as

$$\mathcal{T}\left(\{\boldsymbol{g}_v, v\in\mathcal{V}\}\right) = \underbrace{\frac{\mu^{\tau}}{N}\sum_{v\in\mathcal{V}}\min\left\{0, \tau(\boldsymbol{g}_v)\right\}}_{\text{TNS Target}} + \underbrace{\mu^{\omega}\min_{v\in\mathcal{V}}\omega(\boldsymbol{g}_v)}_{\text{WNS Target}} \quad (8)$$

where μ^{τ} and μ^{ω} are weights for the TNS target and WNS target, respectively. N is the number of gates with negative gatewise TNSs. As the WNS target and TNS target contain minimum operation, directly applying the timing target $\mathcal{T}$ for backward propagation leads to a cut-off in some timing paths. To overcome the drawback, we follow the method proposed in [14] to smooth the minimum and maximum operations. In details, these operations are replaced with the Log-Sum-Exp function as follows:

$$LSE\left(\omega(\boldsymbol{g}_v), v\in\mathcal{V}\right) = \gamma\log\left(\sum_{v\in\mathcal{V}}\exp\frac{\boldsymbol{g}_v}{\gamma}\right) \quad (9)$$

where γ is the critical parameter to adjust the degree of smoothing where a larger γ causes smoother results with lower-approximation accuracy. Similarly, the minimum operation is smoothed by the inverse values. Thus, the value of γ plays an important role in our work to achieve efficient timing optimization and is necessary to be selected carefully. After that, we can get the smoothed $\mathcal{T}(\boldsymbol{g}_v)$. Based on smoothed $\mathcal{T}(\boldsymbol{g}_v)$, the timing optimization gradients w.r.t. gate size ($\nabla_{\boldsymbol{g}_v}\mathcal{T}$) can be computed automatically via backward propagation, which can be used in our gate-sizing framework.

B. Gradient Generation

As shown in (8), the first and fundamental task is to calculate gradients of our timing model w.r.t. gate sizes ($\nabla_{\boldsymbol{g}_v}\tau$ and $\nabla_{\boldsymbol{g}_v}\omega$) before generating gradients of timing target $\nabla_{\boldsymbol{g}_v}\mathcal{T}$. Since gate size $\boldsymbol{g}_v$ of each gate is discrete rather than continuous, the round operation is a necessity in our timing models $\tau(\boldsymbol{g}_v)$ and $\omega(\boldsymbol{g}_v)$ during forward pass. As illustrated in Fig. 6, the continuous gate size $\boldsymbol{g}_v^{\text{con}}$ can be translated into real achievable gate size $\boldsymbol{g}_v$ after it. However, it makes our timing models not differentiable w.r.t. gate sizes. Thus, the sizing-oriented straight-through estimator is developed to solve the issue and generate gradients ($\nabla_{\boldsymbol{g}_v}\tau$ and $\nabla_{\boldsymbol{g}_v}\omega$) accurately for gate sizing. Fig. 6 gives an example of generating $\nabla_{\boldsymbol{g}_v}\tau$ via the sizing-oriented straight-through estimator. In it, the gradient of the round operator is approximated as 1. Based on the approximation, we can get the $\nabla_{\boldsymbol{g}_v}\tau$ as

$$\nabla_{\boldsymbol{g}_v}\boldsymbol{g}_v^{\text{con}} = 1 \rightarrow \nabla_{\boldsymbol{g}_v}\tau = \nabla_{\boldsymbol{g}_v^{\text{con}}}\tau$$
$$\nabla_{\boldsymbol{g}_v}\omega = \nabla_{\boldsymbol{g}_v^{\text{con}}}\omega, \quad v\in\mathcal{V}. \quad (10)$$

This simple approximation function works well in quantization-aware training works. Fortunately, it is also a good method to solve discrete issues in gate sizing-aware timing models. We give an explanation for the efficiency as follows: In quantization works, the float point variables are quantized with bitwise variables. Similar to quantization works, the gate size can be continuous while designing. It is quantized while generating standard libraries to compact the library size and improve design efficiency [1]. Thus, discrete

TABLE I
BENCHMARK STATISTICS. THE UNITS OF "WNS" AND "TNS" ARE NS. AND THE UNIT OF "POW" IS MW

Circuit	#gates	#wires	#CPs	WNS	TNS	NVE	POW
DMX	11616	11671	7068	-0.4112	-2.9417	170	1.4618
GFX	11004	11156	18011	-0.6688	-107.1956	1042	0.9102
AC97	4954	5046	3102	-0.2281	-7.7943	74	1.8342
VGA	32727	32863	10860	-0.8694	-237.1859	2852	4.1925
NOVA	105756	107641	64325	-1.6939	-1982.8487	12007	3.6526
Tot.Train	166057	168377	103366	-3.8714	-2337.9662	16145	12.0513
MC_TOP	3943	4121	5660	-0.2500	-16.7637	216	1.3899
ECG	39992	40941	40963	-0.5941	-771.1540	3830	13.1050
ETH	25327	25450	11681	-1.4086	-401.2948	924	4.2431
USB	6666	7000	6143	-0.4830	-26.1364	93	1.4151
TATE	153192	154720	73147	-1.5241	-2149.5106	5272	29.2070
Tot.Test	229120	232232	137594	-4.2598	-3364.8595	10335	49.3601

issues in gate-sizing work are the same as quantization works. As shown in Fig. 6, the relationship between different sizes can be retained in our work. Based on the generated $\nabla_{\boldsymbol{g}_v}\tau$ and $\nabla_{\boldsymbol{g}_v}\omega$, the timing targets gradients w.r.t. gate sizes $\nabla_{\boldsymbol{g}_v}\mathcal{T}$ can be computed automatically and accurately before backward propagation.

C. Adaptive Gradient Back-Propagation

After obtaining timing target gradients w.r.t. gate sizes $\nabla_{\boldsymbol{g}_v}\mathcal{T}$, the stochastic optimization algorithm proposed in Adam [27] can be applied to optimize the timing target $\mathcal{T}$ via gradient back-propagation. The gate size in our work can be updated as

$$\boldsymbol{g}_v := \boldsymbol{g}_v - \varepsilon_v \nabla_{\boldsymbol{g}_v}\mathcal{T}, \quad v \in \mathcal{V} \tag{11}$$

where ε_v is the learning rate in Adam. However, if we directly perform gradient descent following (11), it is difficult to solve the gate sizing problem with high efficacy on large-scale circuits with many gates. The problem is caused by the high-dimensional issue. When multiple gate sizes change simultaneously, there may be variations in gradient estimation. This is because there are interdependencies among different gates. In the experience of physical designers, it is a common practice to fix some gates while performing gate sizing on others for achieve timing optimization.

In our work, we incorporate the experience of physical designers and use an adaptive learning rate ε_v to update gate sizes based on gradients. If we employ an alternating optimization scheme with sampling, it may result in unacceptable runtime costs. Instead, we utilize the well-known technique of Gumbel-Softmax [28] to achieve adaptive back-propagation via sampling

$$\varepsilon_v = \frac{\exp\big(\big(\log\big(\omega(\boldsymbol{g}_v)\big) + n_v\big)/\lambda\big)}{\sum_{i\in\mathcal{V}} \exp\big(\big(\log\big(\omega(\boldsymbol{g}_i)\big) + n_i\big)/\lambda\big)}, \quad v \in \mathcal{V} \tag{12}$$

where n_v and n_i are independent and identically distributed samples drawn from Gumbel distribution. λ represents the temperature parameter. Our intuition is that since timing issues are determined by their worst-case scenario, we use the normalized result of gatewise WNS $\omega(\boldsymbol{g}_v)$ as the probability value for sampling. For gates with larger gatewise WNS values, which are bottlenecks in timing, more probability is allocated for gradient sampling and gradient back-propagation. It means they should be solved with higher priority.

TABLE II
CHARACTERISTICS OF TSMC 16-NM CELL LIBRARY

Gate Type	Input count options	Gate size options
BUFF	1	0,1,2,3,4,6,8
AND	2,3,4	0,1,2,3,4,8,12,16,20,24
OR	2,3,4	0,1,2,3,4,8,12,16
AOI	3,4,5,6	0,1,2,4,8,12
XOR	2,3,4	0,1,2,4,8
INV	1	0,1,2,3,4,6,8
NAND	2,3,4	0,1,2,3,4,6,8,12,16,20,24
NOR	2,3,4	0,1,2,3,4,8,12,16
OAI	3,4,5,6	0,1,2,4,8,12
XNOR	2,3,4	0,1,2,4,8
MUX	2,3,4	0,1,2,4,8

TABLE III
IMPORTANT SETTINGS USED IN ICC2

Parameters	Detailed Settings
opt. timing. effort	high
opt. common. max_fanout	24
opt. common. max_net_length	120
opt. area. effort	ultra
place. coarse. max$_{\text{d}}$ensity	0
place_opt. congestion. effort	0
refine_opt. congestion. effort	medium
DVAR (pl, clock_derate)	1
DVAR (pl, clock_gating$_{\text{s}}$etup)	0.1
DVAR (pl, clock_uncertainty)	0.25
DVAR (cts, max_clock$_{\text{t}}$ran)	0.06
DVAR (cts, max_clock$_{\text{c}}$ap)	0.15
DVAR (cts, target_skew)	0

VI. EXPERIMENTAL RESULTS

Our framework is implemented in Python with the Pytorch library and in C++. The multimodal timing model is trained on a Linux machine with 32 cores and 4 NVIDIA Tesla V100 GPUs. The training process takes about 4.5 h using the parallel training method on 4 GPUs. The total memory used is 128 GB. In timing target calculation, both the weights for TNS target μ^{τ} and for WNS target μ^{ω} are set to 0.5. And they can be adjusted to meet different timing requirements. To smooth the penalty function described in (9), we set γ as 10.0. The temperature parameter λ used during adaptive gradient back-propagation equals 5.0.

In this work, we train our timing model and evaluate our framework using different open-source designs [30]. And our work can be applied to unseen design without retraining.

TABLE IV
GATEWISE TNS $\tau(g_v)$ AND GATEWISE WNS $\omega(g_v)$ PREDICTION ACCURACY. THE UNIT OF "*MAE*" IS PS

Cir.	DAC22 [29]				DAC23 [23]				Ours W/o Physical Features				Ours			
	$\tau(g_v)$		$\omega(g_v)$		$\tau(g_v)$		$\omega(g_v)$		$\tau(g_v)$		$\omega(g_v)$		$\tau(g_v)$		$\omega(g_v)$	
	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE
DMX	0.75	44.35	0.76	36.14	0.82	38.13	0.87	27.75	0.85	14.28	0.87	11.29	**0.96**	**6.29**	**0.95**	**3.98**
GFX	0.71	36.21	0.74	29.34	0.76	28.15	0.81	22.98	0.82	12.54	0.86	10.89	**0.96**	**5.42**	**0.98**	**3.11**
AC97	0.78	47.56	0. 8	32.68	0.84	35.24	0.86	27.77	0.84	19.12	0.86	16.24	**0.94**	**7.88**	**0.97**	**5.06**
VGA	0.73	56.13	0.78	42.97	0.77	45.23	0.82	36.88	0.86	17.88	0.89	14.25	**0.94**	**6.92**	**0.97**	**3.78**
NOVA	0.79	82.14	0.81	71.54	0.82	50.13	0.84	46.37	0.86	37.28	0.90	23.99	**0.95**	**7.99**	**0.98**	**4.28**
Ave.	0.75	53.28	0.78	42.54	0.81	39.38	0.84	34.43	0.84	20.22	0.88	15.33	**0.95**	**6.50**	**0.97**	**4.04**
TOP	0.67	24.24	0.71	19.15	0.75	17.21	0.78	13.99	0.79	13.75	0.83	8.98	**0.94**	**5.23**	**0.97**	**2.35**
ECG	0.58	37.26	0.64	27.82	0.71	29.12	0.76	27.14	0.79	14.23	0.82	12.59	**0.90**	**4.29**	**0.94**	**3.20**
ETH	0.69	42.75	0.72	34.62	0.77	35.14	0.81	26.32	0.83	11.27	0.87	12.17	**0.93**	**3.97**	**0.95**	**2.12**
USB	0.66	27.99	0.70	15.74	0.76	17.89	0.80	11.24	0.84	8.24	0.89	7.92	**0.92**	**3.12**	**0.94**	**1.27**
TATE	0.70	98.72	0.74	89.22	0.79	57.32	0.85	49.21	0.82	20.28	0.84	16.62	**0.94**	**8.23**	**0.96**	**6.75**
Ave.	0.66	46.19	0.70	37.31	0.76	31.14	0.80	25.58	0.82	13.55	0.85	11.67	**0.94**	**4.77**	**0.95**	**3.14**

TABLE V
TIMING OPTIMIZATION RESULT COMPARISON BETWEEN OUR FRAMEWORK AND OTHER GATE SIZING WORKS

Cir.	ICC2			RL-Sizer [11]			TransSizer [4]			AGD [15]			Ours		
	WNS	TNS	NVE	WNS	TNS	NVE	WNS	TNS	NVE	WNS	TNS	NVE	WNS	TNS	NVE
DMX	-0.1596	-0.9715	92	-0.1465	-0.8521	80	-0.1632	-1.2354	98	-0.1508	-0.9408	89	**-0.1391**	**-0.7998**	**72**
GFX	-0.4129	-30.2528	150	-0.3976	-26.2132	129	-0.4432	-36.2589	192	-0.4021	-36.1854	138	**-0.3659**	**-23.4321**	**98**
AC97	-0.0002	-0.0003	**2**	-0.0003	-0.0012	9	-0.0004	-0.0009	3	-0.0003	-0.0008	4	**-0.0001**	**-0.0002**	**2**
VGA	-0.0104	-0.0456	14	-0.0178	-0.0829	47	-0.0254	-0.0618	32	-0.0162	-0.0745	42	**-0.0093**	**-0.0408**	**12**
NOVA	-0.5214	-28.8674	72	-0.4621	-20.2178	59	-0.5974	-67.2348	348	-0.5438	-35.6218	103	**-0.4339**	**-21.2486**	**31**
TOP	**-0.0002**	-0.0002	**1**	-0.0002	-0.0008	4	-0.0003	-0.0011	8	-0.0003	-0.0013	9	**-0.0002**	**-0.0002**	**1**
ECG	-0.0012	-0.0029	7	-0.0023	-0.0079	26	-0.0017	-0.0051	18	-0.0019	-0.0064	20	**-0.0010**	**-0.0022**	**5**
ETH	-0.1596	-0.9715	92	-0.1465	-0.8521	80	-0.1632	-1.2354	98	-0.1508	-0.9408	89	**-0.1391**	**-0.7998**	**72**
USB	-0.1199	-4.5600	48	-0.1052	-4.0214	42	-0.1256	-5.2365	51	-0.1201	-4.5632	49	**-0.1002**	**-3.514**	**36**
TATE	-0.0013	-0.0055	10	-0.0021	-0.0082	23	-0.0018	-0.0069	15	-0.0019	-0.0072	20	**-0.0011**	**-0.0049**	**8**
Ave.	1.0000	1.0000	1.00	1.1966	1.8236	2.22	1.4055	2.0088	2.52	1.2511	1.9684	2.42	**0.8371**	**0.8139**	**0.78**

The benchmark circuits are synthesized with TSMC 16-nm technology and details are shown in Table I The types and sizes of different cells used in our work are shown in Table II. The circuit benchmarks are split into training and testing sets. The training and testing sets are determined by design scale in order to make balance. The timing evaluation model is trained on the training set with a learning rate of 0.0004. #CPs represent the number of critical paths. WNS and TNS represent the worst and TNS of circuits. NVE represents the number of endpoints with timing violations. POW is the power consumption. We compare our framework with the following advanced baselines: 1) the commercial EDA tool ICC2; 2) RL-sizer [11]; 3) TranSizer [4]; and 4) *AGD [15]:* Timing model proposed in [29]+gradient descent optimization.

A. Baseline Details

For ICC2, there are different settings for achieving timing optimization and meeting physical constraints. The detailed settings are shown in Table III.

TransSizer [4] uses six encoder and decoder layers with eight heads for attention. During training, the learning rate of Adam optimizer is set to 0.0001, the batch size is set to 2048 and the number of epoch is set to 2000. In TransSizer, the size of each gate is predicted path by path. Thus, when a gate appears in more than one path during inference, it picks the sizing solution with the smaller FO4 delay value.

For AGD [15], it runs the optimization process for 300 iterations for each circuit, and the first five iterations are used for fine-tuning. The same training circuits are used to train the timing model [29] used in AGD. The learning rate is set to 0.1 and AdamW optimizer is used for timing optimization. Furthermore, the greedy algorithm is applied on top of the best-sizing solution found by AGD to boost the results further.

B. Timing Model Accuracy

The accuracy of our gate-sizing aware timing model to achieve gatewise TNS and WNS prediction is illustrated in Table IV. Specifically, the R^2 score (the higher the better) and maximum absolute error *MAE* (the lower the better) are used to evaluate the performance. According to the results, they demonstrate that our timing model can accurately predict gatewise TNS and WNS. For the training designs, the average R^2 scores and *MAE* of gatewise TNS and WNS on all gates are 0.95/6.50 and 0.97/4.04 ps. For the unseen testing designs, the average R^2 scores and *MAE* of gatewise TNS and WNS on all gates are 0.94/4.77 and 0.95/3.14 ps. Our proposed framework vastly outperforms all other baseline models on all benchmark circuits for gatewise TNS and gatewise WNS prediction. Compared with DAC22 [29] which collects timing information on one single path, timing information on multiple paths is considered jointly in our work. Compared with DAC23 [23] which collects single-scale physical information, physical information on multiscale layouts is collected in our model. The utilized rich information makes our gate sizing-aware timing model more accurate. The high accuracy helps to improve the timing optimization performance of our work.

C. Timing Performance Improvements

Table V demonstrates the timing optimization results of our work and other comparisons after replacement and rerouting.

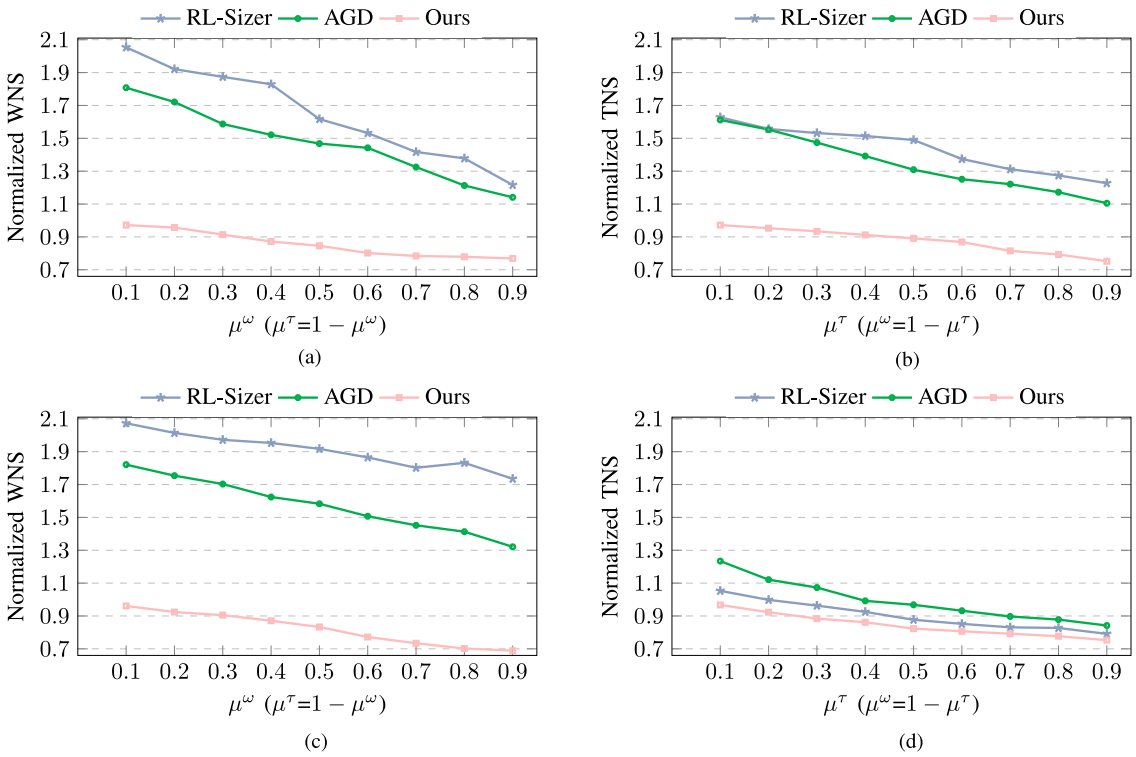

Fig. 7. Normalized TNS and WNS improvement achieved by RL sizer, AGD, and our work across different timing requirements. (a) WNS improvement of TATE. (b) TNS improvement of TATE. (c) WNS improvement of ECG. (d) TNS improvement of ECG.

TABLE VI
RUNTIME COMPARISONS

Circuit	Runtime (min) / Speedup (×)				
	ICC2	RL-sizer [11]	TransSizer [4]	AGD [15]	Ours
DMX	18.72/1.00×	33.27/0.56×	0.65/28.8×	6.23/3.00×	3.12/6.01×
GFX	36.75/1.00×	55.13/0.67×	0.63/58.0×	12.25/3.00×	8.17/4.50×
AC97	7.93/1.00×	14.73/0.54×	0.52/15.4×	2.27/3.50×	1.13/7.00×
VGA	284.97/1.00×	313.07/0.91×	0.97/294.8×	45.15/6.31×	32.12/8.87×
NOVA	153.57/1.00×	342.27/0.45×	1.63/94.0×	41.15/3.73×	28.10/5.47×
TOP	23.45/1.00×	45.88/0.51×	0.37/64.0×	6.12/3.83×	4.08/5.74×
ECG	65.60/1.00×	122.12/0.54×	1.02/64.5×	18.17/3.61×	8.07/8.13×
ETH	22.53/1.00×	47.12/0.48×	0.83/27.0×	7.17/3.14×	3.07/7.35×
USB	17.53/1.00×	31.97/0.55×	0.58/30.1×	4.13/4.24×	3.10/5.66×
TATE	179.37/1.00×	267.55/0.67×	2.15/83.4×	38.08/4.71×	27.05/6.63×
Ave.	81.04/1.00×	127.31/0.59×	0.84/76×	18.07/3.91×	11.80/6.64×

In summary, our framework achieves an average of 16.29% and 18.61% WNS and TNS improvements compared with ICC2. And it also outperforms all other comparisons. After analyzing the results, we summarize our findings below.

1) Our work can achieve gate sizing to optimize timing on seen and unseen circuits. The results suggest that our work can generalize across various designs with different functions and scales without any retraining.
2) We achieve higher-timing performance improvements, including TNS, WNS, and NVE, with ignorable power consumption costs.
3) Compared with RL-sizer, our gradient-based work can achieve more stable optimization.
4) Compared with TranSizer, our work achieves gradient descent optimization. ICC2 results are used as gradient labels rather than classification labels. It helps our work outperform ICC2 rather than imitate it as TranSizer.
5) Compared with AGD, our work achieves better-optimization performance benefiting from the multimodal gate sizing-aware timing model and effective gradient generation and back-propagation.

As described in Section V-A, our work can optimize timing performance according to different requirements. It is achieved by adjusting weights for the TNS target μ^{τ} and WNS target μ^{ω}. Fig. 7 gives results of timing optimization on two Opencore design, including TATE and ECG. They are achieved by RL-sizer [11], AGD [15] and our work when μ^{τ} and μ^{ω} are set to different values which ranges from 0.1 to 0.9. According to the results, we summarize some findings below.

1) Our work outperforms the other two works based on all settings. The results indicate that our work can achieve more stable and efficient optimization across all design spaces.

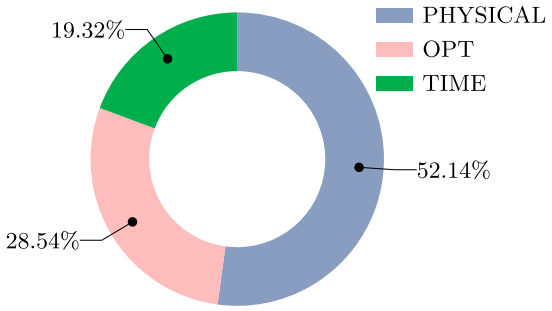

Fig. 8. Runtime breakdown in one gate sizing flow of our work.

TABLE VII
POWER CONSUMPTION RESULT COMPARISONS

Circuit	Power Consumption (mW)				
	ICC2	RL-sizer [11]	TransSizer [4]	AGD [15]	Ours
DMX	2.4312	2.6714	2.3144	2.5284	2.7129
GFX	1.4254	1.4789	1.3542	1.4452	1.4638
AC97	2.3461	2.3456	2.2548	2.2019	2.3502
VGA	6.4087	6.3217	5.3214	6.0257	6.3589
NOVA	5.2132	5.2871	4.8218	4.9687	5.3129
TOP	1.6530	1.4229	1.5478	1.4029	1.6531
ECG	6.7383	6.5587	6.5128	6.4541	6.7891
ETH	2.4312	2.6714	2.2144	2.5284	2.5129
USB	1.5934	1.6354	1.5567	1.5888	1.6512
TATE	37.4630	36.9258	35.6241	34.9887	39.0098
Ave.	1.0000	1.0128	0.9667	1.0235	1.0006

2) Larger μ^{τ} leads to generating circuits with better-TNS optimization, while μ^{ω} leads to better-WNS optimization. It indicates that our work can meet different timing requirements effectively for different applications.

D. Runtime

The time-to-market pressure requires the gate sizing work to be effective on large-scale circuits. The running time of our framework and other comparisons are shown in Table VI. Compared with timing-consuming ICC2 and RL-sizer [11], our work achieves 6.64× and 11.25× speedup, respectively. Compared with TransSizer [4], our work achieves much better-optimization performance in a reasonable time. Compared with other gradient descent optimization works AGD [15], our work achieves acceleration benefiting from optimizing critical paths globally and adaptive back-propagation.

In addition, as demonstrated in Fig. 8, the bulk of runtime in one gate sizing flow—about 70%—is consumed by physical and timing analysis for our work. Thus, the overall runtime is predominantly influenced by the convergence speed of the optimization algorithms. Our framework benefits from an accelerated convergence speed, resulting in faster optimization and more enhanced scalability, which is particularly advantageous for large-scale circuits. This acceleration is achieved through adaptive gradient back-propagation and the trained model based on gradient labels.

E. Power Consumption

The power consumption costs caused by timing optimization are very important for some low-power design. The power consumption results after gate sizing are shown in Table VII. According to the results, our work can achieve more timing performance improvements with less power consumption increment comparing with RL-Sizer [11] and AGD [15]. For TransSizer [4], the timing optimization performance is very poor using similar power consumption costs with other works. Compared with ICC2, the power consumption cost of our work is just 0.06%. The power consumption-efficient timing optimization is achieved based on our trained model using gradient labels. The ignorable costs can be optimized easily by power ECO works [31].

F. Ablation Study

In this section, we conduct ablation studies to demonstrate the effectiveness of our proposed work. We compare the following schemes.

1) *W/o Timing Features:* It can only utilize multiscale physical features via physical feature aggregation proposed in Section IV-C. In the scheme, no timing feature is modeled while achieving gate sizing by our work. Thus, the scheme is very difficult to achieve efficient timing optimization.
2) *W/o Physical Features:* It can only utilize multipath timing features via timing feature aggregation proposed in Section IV-B. Compared with our physically aware work, no physical feature prevents the scheme from capturing gate sizing-induced wire delay variation. Thus, it can not achieve optimal timing performance.
3) *W/o Slack Labels:* It can only utilize gradient labels during gate sizing-aware timing model training. In addition, the predicted gradient labels are directly used during gradient back propagation. In the scheme, no slack label is applied in our loss function in (7). Thus, the scheme can only achieve timing optimization to the same level with ICC2 without further optimization.
4) *W/o Gradient Labels:* It can only utilize slack labels during gate sizing-aware timing model training. In the scheme, no gradient label is applied in our loss function in (7). Thus, the scheme can achieve high-accurate timing prediction without fast optimization gradients.
5) *W/o Gumbel:* It achieves gradient back-propagation without Gumbel-Softmax sampling proposed in Section V-C. Specifically, all size gradients are back-propagated equally in the scheme. Thus, the optimization always falls into local optimal solutions due to gradient vanishing. In addition, more runtime cost is a necessity.
6) *W/o STE:* It achieves gradient generation using the method proposed in AGD [15]. No consideration of relationships among different gate sizes causes inaccuracy during gradient generation. It leads to poor timing optimization performance.
7) *Ours:* It can utilize multiscale layout features and multipath timing features. It achieves gradient back-propagation with Gumbel-Softmax sampling. This scheme is the final implementation of our work.

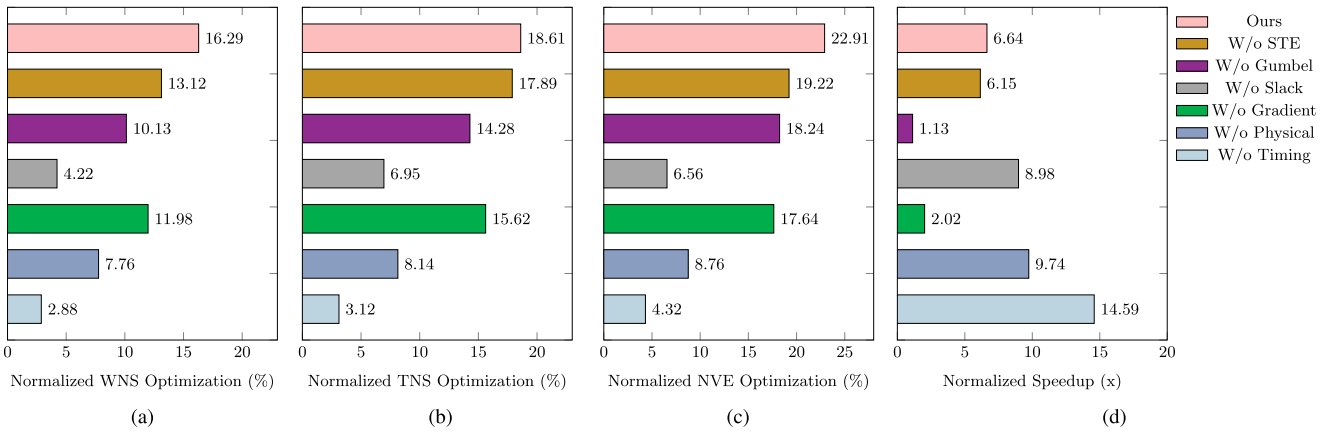

Fig. 9. Comparison among different schemes by (a) WNS optimization, (b) TNS optimization, (c) NVE optimization, and (d) speedup. All these values are normalized by results generated via ICC2.

TABLE VIII
GATEWISE TNS $\tau(\boldsymbol{g}_v)$ AND GATEWISE WNS $\omega(\boldsymbol{g}_v)$ PREDICTION ACCURACY COMPARISON AMONG USING DIFFERENT KINDS OF LABELS. THE UNIT OF "*MAE*" IS PS

Cir.	Ours W/o Slack Labels				Ours W/o Gradient Labels				Ours			
	$\tau(\boldsymbol{g}_v)$		$\omega(\boldsymbol{g}_v)$		$\tau(\boldsymbol{g}_v)$		$\omega(\boldsymbol{g}_v)$		$\tau(\boldsymbol{g}_v)$		$\omega(\boldsymbol{g}_v)$	
	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2	MAE
DMX	0.90	10.35	0.91	8.26	0.91	8.13	0.92	5.87	**0.96**	**6.29**	**0.95**	**3.98**
GFX	0.91	11.24	0.89	9.89	0.93	8.34	0.91	9.27	**0.96**	**5.42**	**0.98**	**3.11**
AC97	0.88	13.58	0.90	11.67	0.90	9.24	0.92	7.77	**0.94**	**7.88**	**0.97**	**5.06**
VGA	0.90	9.38	0.91	8.67	0.91	5.23	0.92	12.02	**0.94**	**6.92**	**0.97**	**3.78**
NOVA	0.92	10.97	0.91	8.26	0.93	8.61	0.91	6.87	**0.95**	**7.99**	**0.98**	**4.28**
Ave.	0.90	11.10	0.90	9.35	0.92	7.91	0.92	8.36	**0.95**	**6.50**	**0.97**	**4.04**
TOP	0.87	10.36	0.87	9.21	0.90	9.65	0.91	8.88	**0.94**	**5.23**	**0.97**	**2.35**
ECG	0.85	12.12	0.89	7.99	0.91	10.23	0.93	9.96	**0.90**	**4.29**	**0.94**	**3.20**
ETH	0.86	13.23	0.91	12.14	0.93	11.33	0.94	9.92	**0.93**	**3.97**	**0.95**	**2.12**
USB	0.84	12.54	0.88	10.54	0.90	10.52	0.92	8.67	**0.92**	**3.12**	**0.94**	**1.27**
TATE	0.85	15.33	0.90	13.64	0.91	13.16	0.93	12.20	**0.94**	**8.23**	**0.96**	**6.75**
Ave.	0.85	12.72	0.89	10.71	0.91	10.98	0.93	9.92	**0.94**	**4.77**	**0.95**	**3.14**

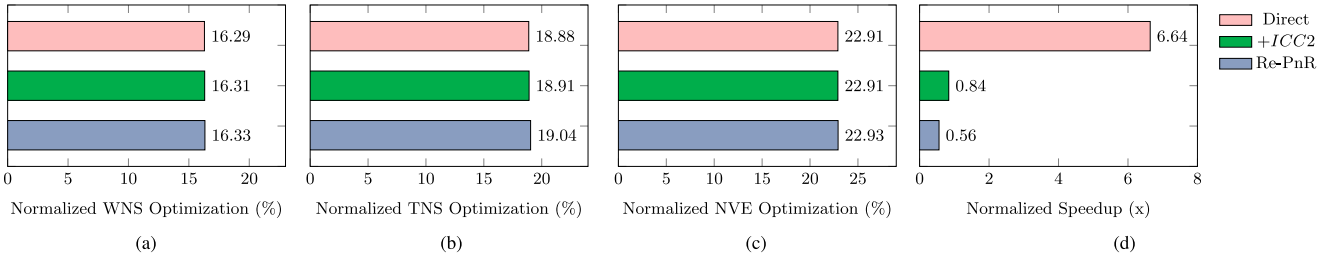

Fig. 10. Comparison among different plans of applying our work by (a) WNS optimization, (b) TNS optimization, (c) NVE optimization, and (d) speedup. All these values are normalized by results generated via ICC2.

As demonstrated in Fig. 9, our work are compared with other works by WNS optimization [see Fig. 9(a)], TNS optimization [see Fig. 9(b)], NVE optimization [see Fig. 9(c)] and speedup [see Fig. 9(d)]. According to our results, the most significant improvement is achieved by aggregating multipath timing features, which is because the timing information on critical paths always is the key to achieve timing optimization. In addition, the multiscale physical aggregated features can also help to enhance the optimization performance by capturing the influence from layouts. Compared with the gradient generation method proposed in AGD [15], the sizing-oriented STE can help our work achieve more timing improvement based on accurate gradient results. These results are generated by capturing relationships between different sizes. As shown in Fig. 9(d), our final work can achieve 6.64× speedup compared with ICC2, which is similar to our work w/o physical features and our work w/o timing features. However, the runtime of our work w/o Gumbel scheme nearly equals the runtime of ICC2. It suggests that adaptive gradient back-propagation through Gumbel-Softmax sampling is efficient to accelerate achieving timing optimization via our work. In summary, the ablation study validates the benefits of using multiscale physical features, multipath timing features, the sizing-oriented straight-through estimator and Gumbel-Softmax sampling.

As demonstrated in Table VIII, we compared our work with its variants without slack labels and gradient labels by

TABLE IX
DISPLACEMENT COMPARISONS. THE UNIT OF MAX DISPLACEMENT AND AVERAGE DISPLACEMENT IS μm

Circuit	ICC2		Ours w/o physical features		Ours	
	Max Displacement	Average Displacement	Max Displacement	Average Displacement	Max Displacement	Average Displacement
DMX	**0.714**	**0.002**	**0.714**	0.005	**0.714**	**0.002**
GFX	**0.990**	0.011	**0.990**	0.015	**0.990**	**0.010**
AC97	**0.754**	**0.001**	**0.754**	0.003	**0.754**	**0.001**
VGA	**0.754**	**0.003**	**0.754**	0.005	**0.754**	**0.003**
NOVA	**0.990**	0.021	**0.990**	0.033	**0.990**	**0.020**
TOP	**1.441**	0.052	**1.441**	0.063	**1.441**	**0.049**
ECG	**1.093**	0.013	**1.093**	0.019	**1.093**	**0.012**
ETH	**1.093**	0.018	**1.093**	0.021	**1.093**	**0.016**
USB	**0.754**	**0.003**	**0.754**	0.004	**0.754**	**0.003**
TATE	**0.700**	**0.001**	**0.700**	0.002	**0.700**	**0.001**
Ave.	**0.928**	0.013	**0.928**	0.017	**0.928**	**0.012**

timing prediction accuracy. Slack labels play important roles in achieving accurate gate sizing-aware timing prediction. However, according to timing optimization results shown in Fig. 9, gradient labels help to achieve faster optimization. Thus, slack and gradient labels are necessities in our work to achieve fast and efficient timing optimization.

As demonstrated in Table IX, we compared our work with its variants without physical features by displacement results after gate sizing. Physical features are critical in our work to achieve timing optimization with less displacement cost. Thus, less displacement-induced wire delay degradation is caused in our work after considering multiscale physical features.

G. Application of Our Method in Different Plans

In this section, we conduct different applications of our method. We compare the following plans.

1) *Direct Application:* It applies our work directly. The plan achieves a good tradeoff between timing optimization efficiency and performance.
2) *Include Re-PnR:* It applies our work with repeating the replacement and rerouting in the flow. After timing-consuming timing and physical analysis, related timing and physical features used in timing optimization can be updated in the plan. These accurate updated features help the plan to obtain more timing improvements.
3) *Ours*+ICC2*:* It combines our work and ICC2 through applying our work at first, then applying ICC2 to achieve further optimization. For the traditional computational optimization method used in ICC2, a good initial solution is critical to achieve optimal timing optimization. Thus, in this plan, we try to improve the timing performance of ICC2 by using the generated result of our learning-driven work as the initial solution.

As demonstrated in Fig. 10, our work is compared with other works by WNS optimization [see Fig. 10(a)], TNS optimization [see Fig. 10(b)], NVE optimization (see Fig. 10(c) and speedup [see Fig. 10(d)]. According to our results, the most significant improvements in different applications are achieved by our work. During the training process of our gate-sizing aware timing model, we extract physical features before replacement and rerouting to achieve timing prediction after gate-sizing. The trained model can accurately capture the gate sizing-induced delay optimization based on original physical features. Thus, our work can obtain similar timing performance improvements compared with repeating the replacement and rerouting in the flow. However, as shown in Fig. 10(d), the runtime cost of timing analysis and physical analysis after repeating the replacement and rerouting limits the optimization efficiency. In addition, the ICC2 can only achieve an average 0.01% timing improvement after our work.

VII. CONCLUSION

This work proposes and implements a learning-driven physically aware gate sizing framework to achieve timing optimization on large-scale circuits efficiently. The powerful and efficient optimization is from the following.

1) Modeling timing optimization and degradation caused by gate-sizing accurately in a multimodal way via learning timing information on multiple timing paths and physical information on multiple scaled layouts.
2) Generating and back-propagating gradients efficiently to update gate sizes via sizing-oriented straight-through estimator and adaptive sampling. Experimental results on open-source designs show that our work can achieve 16.29% and 18.61% TNS and WNS improvements on average compared with the commercial gate sizing tool. In addition, it obtains a 6.64× speedup.

REFERENCES

[1] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Dordrecht, The Netherlands: Springer, 2011, vol. 312.
[2] A. B. Kahng, S. Kang, H. Lee, I. L. Markov, and P. Thapar, "High-performance gate sizing with a signoff timer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2013, pp. 450–457.
[3] (Synopsys, Sunnyvale, CA, USA). *IC Compiler II User Guide*. (2023). [Online]. Available: https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html
[4] S. Nath, G. Pradipta, C. Hu, T. Yang, B. Khailany, and H. Ren, "TransSizer: A novel transformer-based fast gate sizer," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2022, pp. 1–9.
[5] V. S. Livramento, C. Guth, J. L. Guentzel, and M. O. Johann, "A hybrid technique for discrete gate sizing based on lagrangian relaxation," *ACM Trans. Design Autom. Electron. Syst. (TODAES)*, vol. 19, no. 4, pp. 1–25, 2014.
[6] A. Sharma, D. Chinnery, S. Bhardwaj, and C. Chu, "Fast lagrangian relaxation based gate sizing using multi-threading," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2015, pp. 426–433.
[7] A. Sharma, D. Chinnery, T. Reimann, S. Bhardwaj, and C. Chu, "Fast Lagrangian relaxation-based multithreaded gate sizing using simple timing calibrations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. (TCAD)*, vol. 39, no. 7, pp. 1456–1469, Jul. 2020.

[8] S. Roy, D. Liu, J. Um, and D. Z. Pan, "OSFA: A new paradigm of gate-sizing for power/performance optimizations under multiple operating conditions," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2015, pp. 1–6.
[9] D. Mangiras, D. Chinnery, and G. Dimitrakopoulos, "Task-based parallel programming for gate sizing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 4, pp. 1309–1322, Apr. 2023.
[10] S. Daboul, N. Hähnle, S. Held, and U. Schorr, "Provably fast and near-optimum gate sizing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3163–3176, Dec. 2018.
[11] Y.-C. Lu, S. Nath, V. Khandelwal, and S. K. Lim, "RL-sizer: Vlsi gate sizing for timing optimization using deep reinforcement learning," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 733–738.
[12] C.-K. Cheng, C. Holtz, A. B. Kahng, B. Lin, and U. Mallappa, "DAGSizer: A directed graph convolutional network approach to discrete gate sizing of VLSI graphs," *ACM Trans. Design Autom. Electron. Syst.*, vol. 28, no. 4, pp. 1–31, 2023.
[13] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu, "Global placement with deep learning-enabled explicit routability optimization," in *Proc. IEEE/ACM Design, Autom. Test Eur. (DATE)*, 2021, pp. 1821–1824.
[14] S. Liu, Z. Wang, F. Liu, Y. Lin, B. Yu, and M. Wong, "Concurrent sign-off timing optimization via deep steiner points refinement," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.
[15] P. Pham and J. Chung, "AGD: A learning-based optimization framework for EDA and its application to gate sizing," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.
[16] G. Chen, Z. Wang, B. Yu, D. Z. Pan, and M. D. Wong, "Ultra-fast source mask optimization via conditional discrete diffusion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 7, pp. 2140–2150, Jul. 2024.
[17] B. Zhu et al., "L2O-ILT: Learning to optimize inverse lithography techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 3, pp. 944–955, Mar. 2023.
[18] Z. Guo and Y. Lin, "Differentiable-timing-driven global placement," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, 2022, pp. 1315–1320.
[19] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–30.
[20] H. Le, R. K. Høier, C.-T. Lin, and C. Zach, "AdaSTE: An adaptive straight-through estimator to train binary neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 460–469.
[21] Z. Yang, J. Lee, and C. Park, "Injecting logical constraints into neural networks via straight-through estimators," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 25096–25122.
[22] (Synopsys, Sunnyvale, CA, USA). *StarRC User Guide*. (2023). [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/starrc.html
[23] Z. Wang, S. Liu, Y. Pu, S. Chen, T.-Y. Ho, and B. Yu, "Restructure-tolerant timing prediction via multimodal fusion," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.
[24] (Synopsys, Sunnyvale, CA, USA). *PrimeTime User Guide*. (2023). [Online]. Available: https://www.synopsys.com/cgi-bin/imp/pdfdla/pdfr1.cgi?file=primetime-wp.pdf
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
[26] R. Azad, M. Asadi-Aghbolaghi, M. Fathy, and S. Escalera, "Attention deeplabv3+: Multi-level context attention mechanism for skin lesion segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 251–266.
[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[28] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–13.
[29] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-nrouting slack prediction," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2022, pp. 1207–1212.
[30] "OpenCores." 2023. [Online]. Available: http:///opencores.org/
[31] Y.-C. Lu, S. Nath, S. S. K. Pentapati, and S. K. Lim, "A fast learning-driven signoff power optimization framework," in *Proc. 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.

Yuyang Ye received the Ph.D. degree from Southeast University, Nanjing, China, in 2024.

He is currently a Postdoctoral Researcher with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His current research interests include machine learning for EDA, timing analysis, and optimization.

Dr. Ye received the Best Student Paper Award from ICSICT 2022.

Peng Xu received the B.S. degree from Central South University, Changsha, China, and the M.S. degree from the Harbin Institute of Technology (Shenzhen), Shenzhen, China, in 2019 and 2021, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. Bei Yu.

His research interests include machine learning for analog physical design and optimization in EDA problems.

Lizheng Ren received the B.S. and M.S. degrees from Southeast University, Nanjing, China, in 2013 and 2016, respectively, where he is currently pursuing the Eng.D. degree.

He is also affliated with Nanjing Low Power IC Technology Institute, Nanjing, as a Staff Engineer. His current research interests include physical design of integrated circuits, and electronic design automation of conventional and emerging VLSI technologies.

Tinghuan Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, respectively, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2021.

He is currently an Assistant Professor with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China. His research interests include machine learning for EDA and deep learning accelerators.

Hao Yan (Member, IEEE) received the B.S. degree from the Dalian University of Technology, Dalian, China, in 2011, and the M.S. and Ph.D. degrees from Southeast University, Nanjing, China in 2014 and 2018, respectively.

He is currently an Associate Professor with the National ASIC Research Center, Southeast University. His research focuses on design methodology for wide-voltage and high-efficiency design, including timing analysis and optimization.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received 11 Best Paper Awards from ICCAD 2024 and 2021 and 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 and 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, the IEEE CEDA Ernest S. Kuh Early Career Award in 2021, the DAC Under-40 Innovator Award in 2024, and The Hong Kong RGC Research Fellowship Scheme (RFS) Award in 2024. He has served as a TPC Chair for ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.

Longxing Shi (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Southeast University, Nanjing, China, in 1984, 1987, and 1992, respectively.

From 1992 to 2000, he was an Associate Professor with the School of Electronic Science and Engineering, Southeast University, where has been a Professor and the Dean of the National ASIC Research Center since 2001. He has authored one book and over 130 articles. His current research interest includes ultralow power IC design and design methodology.