Addressing Continuity and Expressivity Limitations in Differentiable Physical Optimization: A Case Study in Gate Sizing

Yufan Du[1,2], Zizheng Guo[2,3], Yang Hsu[5], Zhili Xiong[5], Seunggeun Kim[5],
David Z. Pan[5], Runsheng Wang[2,3,4], Yibo Lin[2,3,4*]
[1]School of EECS, Peking University, [2]School of Integrated Circuits, Peking University, [3]Institute of EDA, Peking University,
[4]Beijing Advanced Innovation Center for Integrated Circuits, [5]University of Texas at Austin
nbsdyf@stu.pku.edu.cn, {yanghsu, zhilix691, sgkim}@utexas.edu,
dpan@ece.utexas.edu, {gzz, r.wang, yibolin}@pku.edu.cn

***Abstract*—Differentiable optimization is popular for its efficiency and explainability. However, it faces limitations due to its reliance on continuous formulations and constraints on objective expressivity. To address these challenges, we propose a framework combining differentiable methods with gradient clipping and calibration strategies to ensure efficient and targeted optimization. Gate sizing, a key challenge in chip PPA optimization, exemplifies all the challenges with its discrete nature and objective complexity. Applying our proposed differentiable framework to gate sizing, we outperform top contestants in the 2024 ICCAD CAD gate sizing contest in overall quality scores and runtime, with excellent and balanced performance on all important evaluation metrics.**

***Index Terms*—physical design, differentiable optimization, gate sizing**

I. Introduction

Differentiable optimization has shown remarkable performance in tackling complex optimization tasks, especially in VLSI design [1]–[8]. By formulating optimization objectives in a differentiable manner, this approach facilitates direct gradient descent optimization resembling the neural network training process. With GPU acceleration, it addresses large-scale problems efficiently [9], [10].

However, applying differentiable optimization to real-world physical design applications poses significant challenges due to inherent *continuity* and *expressivity* limitations. Encountering these limitations, we proposed respective solutions. To verify our solutions, we conduct a case study in gate sizing—a critical and representative VLSI optimization task. Gate sizing selects gate driving strengths to optimize timing, power, and area, subjecting to design rule constraints. It is a crucial step in the design flow for PPA (power, performance, and area) optimization since it is frequently triggered after key stages like placement, clock tree synthesis, and routing.

Like other optimization tasks, gate sizing in prior works [5]–[7] has benefited from the remarkable efficiency of differentiable optimization methods but struggles with the aforementioned limitations. The *continuity* limitation [11], [12] poses a significant challenge, as almost all core VLSI tasks—such as logic optimization, placement, and routing—require discrete solutions that conflict with the continuous nature of differentiable frameworks. Particularly in gate sizing, the extremely sparse gate size solutions are highly incompatible with traditional differentiable methods like gradient descent.

On the other hand, the *expressivity limitation* [13], [14] stems from the necessity of using smooth models to enable differentiable computation. However, many timing and power models in VLSI design are inherently not smooth. Since different models yield different analyzing outcomes, and the choice of the final outcome ultimately lies with the user, optimization efforts must be reflected regardless of the selection of the reference model. For instance, in nearly all VLSI design tasks, including gate sizing, timing is a critical objective. Yet, it is widely recognized that timing models suffer from the *expressivity limitation*. Previous works [3], [5]–[7] only optimize timing in their own differentiable timing models. However, once the reference timing model alters, their optimization effects are mistargeted.

Since both limitations of a differentiable framework are very common across almost all physical optimization tasks, the need for general methods to solve these limitations is urgent. In our work, we propose strategies to overcome these limitations and develop a novel differentiable framework using gate sizing as a case study. Following are our contributions:

- To tackle the *continuity* limitation, we propose a *gradient clipping* strategy for gate sizing discretization that bridges the gap between the continuous differentiable optimization and the discrete nature of the gate sizes.
- To address the *expressivity* limitation, we introduce a *gradient calibration* framework in which our optimization efforts are aimed regardless of the selection of the external reference VLSI analyzing tool.
- Our experimental results show that our proposed differentiable framework performs exceptionally well in the gate sizing case study in all important metrics. We outperform all top 3 winners at 2024 ICCAD CAD gate sizing contest in both quality score and runtime.

We believe that our work will inspire and encourage further research on efficient differentiable physical optimization. Our advancements could broaden the scope of optimization in chip design and other real-world applications.

II. Preliminaries

This section reviews differentiable optimization techniques for physical design tasks and introduces the gate sizing problem.

A. Differentiable Optimization

Differentiable optimization has recently gained wide popularity for addressing complex optimization challenges in VLSI design fields [1]–[8], [15]. The key idea is to formulate the variable into a continuous solution space (e.g., by relaxing discrete constraints). Then, the optimization objective is expressed as a differentiable analytical function whose gradient is derived using back-propagation and used directly for gradient-based optimization.

* Corresponding author.

This work was done in part while Yufan Du was a visiting student at UT Austin during the summer of 2024.

TABLE I: Summary of notations.

Notation	Description
$\mathcal{G}$	Set of gates
$\mathcal{S}_g$	Set of available sizes for gate g
$\mathcal{E}$	Set of edges in the timing graph
$\mathcal{N}_{\text{end}}$	Set of end nodes in the timing graph
$\mathbf{s}_g$	Size for gate g
$d_{ij}(\mathbf{s}_g)$	Delay from node i to node j
a_i	Arrival time at node i
t	Clock period
t_{setup}	Setup time
$\text{Leak}_g(\mathbf{s}_g)$	Leakage power of gate g with size $\mathbf{s}_g$
$\text{Area}_g(\mathbf{s}_g)$	Area of gate g with size $\mathbf{s}_g$
$\text{Cap}_i(\mathbf{s}_g)$	Capacitance of the pin i with its corresponding gate size $\mathbf{s}_g$
$\text{Load}_i(\mathbf{s})$	Load capacitance at output pin i
$\text{Slew}_i(\mathbf{s})$	Transition time at input pin i
TNS	Total negative slack
WNS	Worst negative slack

Differentiable optimization has been used to optimize cell placement for better timing [3], signal routing for better wirelength and routability [15], approximate logic synthesis solutions for better area and error rate [2] and neural architecture search (NAS) [16]. Such differentiable techniques benefit from an inherently well-formed solution space and a more targeted and explainable optimization objective. As a result, they have been showing superior potential.

B. Case Study: Gate Sizing

Gate sizing selects the drive strength of each gate to optimize timing and eliminate DRVs like maximum load capacitance and signal transition time. This process ensures gates on critical paths have enough strength to pass signals efficiently while reducing non-critical gate sizes to save power and area. As such, gate sizing becomes a critical factor connecting all important chip quality metrics.

As an NP-hard problem [17], gate sizing is among the most difficult combinatorial optimization problems. Specific challenges arise from its discrete nature, the non-convexity of delay models, and the large number of near-critical paths that make optimization challenging [18]. Prior research on gate sizing generally falls into the following categories:

1) *Dynamic programming-based methods* [19]–[21] work well for tree-structured circuits but falter with general circuits containing reconvergent paths.
2) *Sensitivity-based methods* [22]–[24] adjust gate sizes based on initial sensitivity estimates, which relies heavily on the quality of heuristics.
3) *Learning-based methods*, including RL [25], generative AI [26], and GCN [27], [28], incur time-consuming retraining when transferred to different technology libraries [29].
4) *Heuristic methods with Lagrangian relaxation (LR)* [30]–[38] reduce the search space using KKT conditions but often rely on slow, local searches and gate-by-gate iterations.

Recently, *differentiable methods* [5]–[7] act as a powerful methodology for gate sizing. While they efficiently optimize PPA metrics using gradient-based approaches, their discrete size of gates mismatches with continuous gradient descent method (*continuity* limitation). Meanwhile, such techniques only guarantee their optimization efforts in their own analyzing model outcomes [6], [7] (*expressivity* limitation), wasting the optimization resources and efforts if an external reference timer is used.

C. Problem Formulation

Problem. Given a set of gates and detailed placement layout, determine the size $\mathbf{s}$ of all gates in order to minimize total leakage power while eliminating DRVs and timing violations. Using the notations from Table I, the gate sizing problem can be formulated as follows:

$$
\begin{aligned}
\min_{\mathbf{s},a} \quad & \sum_{g\in\mathcal{G}} \text{Leak}_g(\mathbf{s}_g) \\
\text{subject to} \quad & \mathbf{s}_g \in \mathcal{S}_g, && \forall g \in \mathcal{G}, \\
& \text{Slew}_i(\mathbf{s}_g) \le \text{Max Slew}_i(\mathbf{s}_g), && \forall i \in \text{Input Pin}(\mathbf{s}), \\
& \text{Load}_i(\mathbf{s}_g) \le \text{Max Load}_i(\mathbf{s}_g), && \forall i \in \text{Output Pin}(\mathbf{s}), \\
& a_i + d_{ij}(\mathbf{s}) \le a_j, && \forall (i,j) \in \mathcal{E}, \\
& a_k \le t - t_{\text{setup}}, && \forall k \in \mathcal{N}_{\text{end}},
\end{aligned}
$$

where minimization is over the set of discrete gate variable $\mathbf{s}$ and arrival time a. There are three constraints:

1) Gate sizes $\mathbf{s}_g$ must be selected from the available sizes $\mathcal{S}_g$ for each gate g.
2) Signal transition time (slew) and gate output load capacitance must strictly stay within upper limits to ensure the elimination of DRVs. This is paramount, as failing to address DRVs directly undermines the reliability of timing analysis, rendering untrustworthy results [39], [40].
3) Once DRVs are eliminated, the setup time requirement must be met to guarantee the circuit's proper functionality.

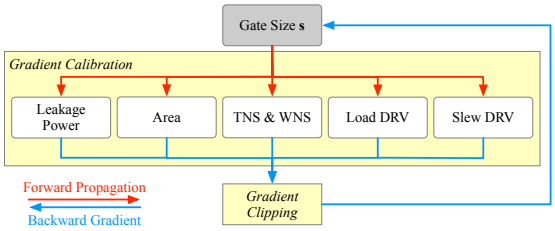

Fig. 1: The flow of our proposed methodology.

III. Methodology

In this section, we present a new differentiable framework. Section III-A provides an overview of our framework. Sections III-B and III-C presents our basic differentiable formulation. Sections III-D and III-E point out the specific *continuity* and *expressivity* limitations and our solutions, respectively.

A. Framework Overview

In a differentiable framework, we first calculate optimization objectives. In our case study, formulating gate sizing objectives is no exception (III-B and III-C). Here is the overall objective, including leakage power consumption, timing metrics, and DRV violation:

$$
\begin{aligned}
\min_{\mathbf{s}} \quad & \text{Leak}(\mathbf{s}) + \alpha|\text{TNS}| + \beta \sum_{\text{pin}_i} \max(\text{Slew}_i(\mathbf{s}) - \text{Max Slew}_i(\mathbf{s}), 0) \\
& + \gamma \sum_{\text{pin}_i} \max(\text{Load}_i(\mathbf{s}) - \text{Max Load}_i(\mathbf{s}), 0),
\end{aligned}
$$

where α, β, and γ are weights for different objectives. These weights are set based on contest factors used to calculate the quality score.

After objective calculation, the gradients of the objective with respect to optimization variable $\mathbf{s}$ can be calculated, indicating how the optimization variable $\mathbf{s}$ should be optimized.

As mentioned before, this basic framework poses significant challenges, particularly regarding *continuity* and *expressivity*. To address the *continuity* limitation, we propose a *gradient clipping* method in III-D to bridge the gap between differentiable framework and the discrete nature of practical tasks. To solve the *expressivity* limitation,

we develop a *gradient calibration* strategy in III-E to enhance optimization target precision regardless of the reference model selection. The whole flow is shown in Figure 1.

B. Differentiable Power and Area Objectives

Power and area objectives are critical in chip design. While cell upsizing can reduce DRV and timing violations, it increases power consumption, leading to heat and reliability issues, larger area usage, and routability difficulties. Balancing these trade-offs while optimizing timing is known to be highly complex. Thus, modeling how gate size changes affect these objectives is essential for targeted and efficient upsizing.

To enable the differentiability, we define a general form of linear interpolation as:

$$F_{\ Interp.}(\mathbf{s}_g) = F(\lfloor \mathbf{s}_g \rfloor)(\lceil \mathbf{s}_g \rceil - \mathbf{s}_g) + F(\lceil \mathbf{s}_g \rceil)(\mathbf{s}_g - \lfloor \mathbf{s}_g \rfloor), \quad (1)$$

where $\mathbf{s}_g$ is gate g size and F is a specific function[2]. This linear interpolation method gives the correct output for each gate size input while ensuring the output is differentiable with respect to the gate size input, thus enabling both objective and gradient calculation.

Applying this to gate sizing, the leakage and area for non-integer gate sizes $\mathbf{s}_g$ are given by:

$$\text{Leak}_g(\mathbf{s}_g) = \text{Leak}_{g\ Interp.}(\mathbf{s}_g), \quad (2)$$

$$\text{Area}_g(\mathbf{s}_g) = \text{Area}_{g\ Interp.}(\mathbf{s}_g). \quad (3)$$

Total power is then summed directly, while the area calculation incorporates local density to consider possible congestion issues:

$$\text{Leak}(\mathbf{s}) = \sum_g \text{Leak}_g(\mathbf{s}_g), \quad (4)$$

$$\text{Area}(\mathbf{s}) = \sum_g \text{Area}_g(\mathbf{s}_g) \times \text{Density}(g). \quad (5)$$

Each gate receives gradients from these objectives:

$$\frac{\partial}{\partial \mathbf{s}_g}\text{Leak}(\mathbf{s}) = \text{Leak}_g(\lceil \mathbf{s}_g \rceil) - \text{Leak}_g(\lfloor \mathbf{s}_g \rfloor), \quad (6)$$

$$\frac{\partial}{\partial \mathbf{s}_g}\text{Area}(\mathbf{s}) = [\text{Area}_g(\lceil \mathbf{s}_g \rceil) - \text{Area}_g(\lfloor \mathbf{s}_g \rfloor)] \times \text{Density}(g). \quad (7)$$

This approach discourages upsizing that consumes high power or occurs in dense regions, effectively controlling heat, congestion, and routing issues.

C. Differentiable DRV and Timing Objectives

Eliminating DRVs and timing violations are complex and critical objectives in VLSI. DRV elimination ensures the reliability of timing analysis, while timing optimization guarantees circuit functionality.

Here, we outline the steps to calculate these complex and critical objectives while maintaining a differentiable manner.

1) Steiner Tree Construction: We utilize FLUTE [41] for early net routing estimates, fed into the timing model as shown in the upper left gray box in Figure 2.

[2]If $\mathbf{s}_g$ is an integer, these two values are the same. To prevent this, we redefine roundings as $\lceil \mathbf{s}_g \rceil = \min(\mathbf{s}_{g\,\max}, \lceil \mathbf{s}_g + \epsilon \rceil)$, where ϵ is a small constant ensuring different rounding values, and $\mathbf{s}_{g\,\max}$ is the maximum gate size. If $\mathbf{s}_g$ is already at $\mathbf{s}_{g\,\max}$, $\lfloor \mathbf{s}_g \rfloor$ becomes $\max(1, \mathbf{s}_{g\,\max} - 1)$ and $\lceil \mathbf{s}_g \rceil$ remains $\mathbf{s}_{g\,\max}$.

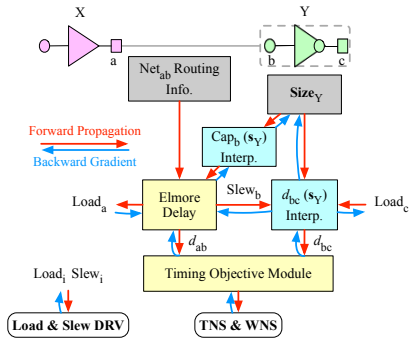

Fig. 2: Computation and gradient flow for differentiable DRV and timing objectives. Slew propagation is omitted for simplicity.

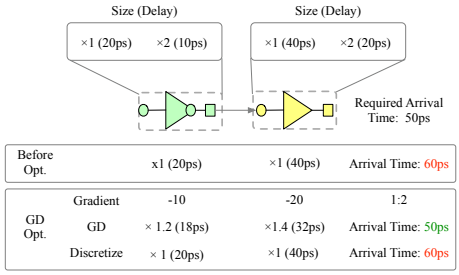

Fig. 3: *Continuity* limitation for natural discrete optimization task in a differentiable framework.

2) Differentiable Gate Input Pin Capacitance: Upsizing a gate typically increases input pin capacitance, impacting its upstream components' timing performance. Therefore, we interpolate the capacitance value for non-integer gate sizes, as shown in the middle blue box. This is done using the following interpolation formula:

$$\text{Cap}_i(\mathbf{s}_g) = \text{Cap}_{i\ Interp.}(\mathbf{s}_g), \quad (8)$$

where pin_i is gate g input.

3) Differentiable Net Delay Calculation: We implement Elmore delay for differentiable delay and slew calculations as in [3] with inputs from the last two steps, as shown in the Elmore Delay box.

4) Differentiable Gate Delay Calculation: We derive delay and slew using a look-up table (LUT) indexed by input slew and output load, with interpolation for non-integer gate sizes (illustrated in the lower right blue box). Specifically:

$$d_{ij}(\mathbf{s}_g) = d_{ij\ Interp.}(\mathbf{s}_g), \quad (9)$$

where i and j denote pins in a timing arc within gate g.

5) Differentiable DRV, WNS, and TNS Calculation: Given capacitance, slew, and delay outputs, we calculate DRV value and timing objectives, as shown in the bottom part of the figure.

D. Gradient Clipping

In previous sections, we constructed differentiable targets. Therefore, each gate can derive the gradient from various objectives. However, effectively leveraging these gradients for optimization remains a significant challenge.

In this section and subsequent sections, we outline the challenges and propose solutions to address them.

1) Continuity Limitation: While gradient descent (GD) in previous work [5] is effective for continuous optimization, it faces limitations when applied to inherently discrete tasks such as gate sizing, making applying differentiable methods to prevailing discrete problems

highly non-trivial. For example, in Figure 3, consider an example where two gates are initially sized at $\times 1$. GD optimization may result in continuous sizes like $\times 1.2$ and $\times 1.4$, that, when interpolated, resolve timing violations. However, since gate sizes are inherently integers, rounding these continuous values to discrete sizes in the final step will cancel out the optimization. With a deeper circuit logic level, this inaccuracy would be amplified [5].

2) Gradient Clipping Solution: Unlike direct usage of GD in previous work [5], we propose a *gradient clipping* strategy to adapt differentiable strategies for discrete optimization problems.

Our algorithm enforces integer gate sizes to preserve the discrete nature, ensuring compatibility with discrete optimization tasks. We leverage gradients as reliable sensitivity indicators to identify the criticality of each optimization variable's adjustment. In our gate sizing case study, initially, all gate sizes are set to minimal. In each iteration, we upsize the top $k_1\%$ gates with the smallest gradients, as these gates will most likely benefit from adjustments. Conversely, to mitigate unnecessary area and power consumption, our algorithm also downsizes the top $k_2\%$ gates with the largest gradients, which are supposed to be oversized gates. The selection of k_1 and k_2 is crucial to balance the convergence speed and optimization quality. Our empirical observations over all test cases show that $0.3 \leq k_1 \leq 1$ and $k_2 \leq 0.1$ strike an effective balance, ensuring efficient optimization without compromising overall quality.

This *gradient clipping* strategy aligns with the discrete constraints of gate sizing while utilizing gradient-based insights for effective adjustments, ensuring robust and practical optimization.

E. Gradient Calibration

In addition to the *continuity* limitation, *expressivity* limitation also restrains the potential of differentiable optimization.

1) Expressivity Limitation: Ensuring compatibility with differentiable optimization methods often necessitates differentiable models. For instance, [5] employs the Elmore delay model to achieve differentiability, but it only maximizes the optimization efforts solely on Elmore delay model. Similarly, [6] models only the effect of gate size on the current gate delay to maintain differentiability, neglecting its impact on upstream load and downstream slew. Furthermore, [7] utilizes machine learning to fit timing models and leverages back-propagated gradients for gate size optimization. However, machine learning models often fall short in accurately replicating algorithmic precision [13], [14].

Achieving both expressivity and differentiability presents a significant challenge. Prior methods [42], [43] have aimed to calibrate objectives to enhance expressivity and accuracy, typically involving additive corrections to modify the original model outputs to align with commercial STA engines. However, such additive adjustments are zero-order, as they do not alter the gradient. Consequently, while these methods ensure correctness in the forward pass, they fail to calibrate the backward gradient calculation.

2) Multiplicative Gradient Calibration Solution: To address the *expressivity limitation*, we introduce *gradient calibration*, a crucial step for ensuring both accurate objective and gradient values. Unlike additive calibration in previous works, we design multiplicative calibration, which introduces a calibration ratio to scale the original value to align any external model.

In our case study, we use timing and design rule violation (DRV), the most significant yet impacted by expressivity limitation metrics, to demonstrate our multiplicative calibration. For these objective calculations, since the slew value propagates throughout all circuit levels, the slew value misalignment is amplified through the propagation

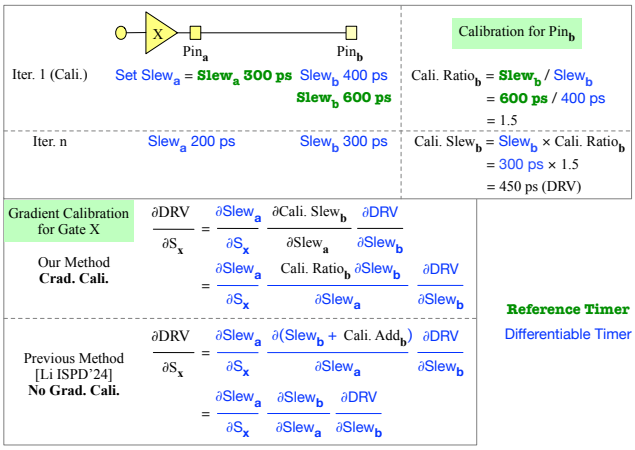

Fig. 4: Demonstration of slew gradient calibration. Calibration for other metrics follows a similar process.

process. Therefore, we highlight the slew calibration. As illustrated in Figure 4, consider a simplified case where gate X drives a 2-pin net. To calibrate the slew value, our timer uses the source Pin_a slew value from the reference timer, followed by both timers computing the sink Pin_b timing metrics. If the slew at Pin_b is reported as 400 ps by our timer and 600 ps by the reference timer, a calibration ratio of 1.5 is derived and used in subsequent iterations to adjust our model's sink pin slew. This approach extends to net delay, net load capacitance, and gate power, enabling comprehensive calibration.

Unlike previous additive calibration methods [42], [43], we find that multiplicative calibration better captures the relationship between variables and objectives. First, the multiplicative method calibrates gate size sensitivity, as reflected in the gradient, leading to a more explainable objective. Second, as observed in our experiments, the multiplicative method remains accurate over extended periods, even as the design undergoes multiple optimization iterations, whereas simple additive calibration diverges quickly. Therefore, in our case study, only a single invoke of multiplicative calibration at the beginning proves to be both efficient and accurate enough.

Notably, the multiplicative calibration approach is a general idea. It can be easily adapted to other cases requiring the alignment between an optimization model and an external model.

IV. Experimental Results

A. Experimental Setup

We implement our proposed gate sizer using Python, C++, and CUDA kernels. To ensure a fair comparison against other contestants in the 2024 ICCAD CAD gate sizing contest [18], we designed our evaluation environment to match the contest conditions. The evaluation machine featured a 2.25 GHz AMD EPYC 7742 processor, 512 GB of RAM, and an Nvidia A100 GPU with 80 GB of memory, consistent with the contest's GPU configuration. In adherence to the contest's constraints, our evaluations were conducted using only 8 CPU cores and a maximum of 200 GB of RAM.

B. Results and Analysis

We evaluated our newly developed gate sizer against the top 3 winners from the contest over all 10 benchmarks. Detailed statistics for these benchmarks are provided in Table II, highlighting the diversity in scale, complexity, power consumption, and timing violation severity across the cases, thereby covering a broad spectrum of design scenarios. These benchmarks are at the post-placement stage, where

TABLE II: 2024 ICCAD CAD gate sizing contest [18] benchmark statistics.

Design	#Gate	WNS (ns)	TNS (ns)	Slew DRV (ns)	Load DRV (fF)	Power (uW)
NV_NVDLA_partition_m	27,553	-0.595	-156.323	258.761	256	1.672
NV_NVDLA_partition_p	79,919	-1.519	-6,306.640	6,125.512	5,292	5.539
ariane136	145,776	-1.298	-10,143.711	14,843.895	15,463	17,539.095
mempool_tile_wrap	187,851	-1.315	-10,458.099	12,069.070	10,779	2,590.189
aes_256	278,465	-0.284	-212.965	942.810	1,300	16.771
hidden1	38,089	-1.069	-1,136.054	4,073.071	6,811	2.762
hidden2	149,396	-1.214	-9,400.457	16,582.889	16,661	17,152.379
hidden3	184,863	-1.563	-2,436.288	19,755.937	33,088	16,513.594
hidden4	260,483	-3.185	-25,334.022	19,138.199	27,548	21.024
hidden5	283,750	-0.324	-370.293	3.483	0	16.170

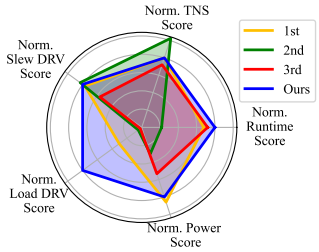

Fig. 5: A holistic radar chart for four sizers. Outer is better.

TABLE III: Gate sizer scores across 10 cases for top three winners, three ablation studies, and our sizer. The best scores are **bolded**, and the second-best scores are highlighted among the top three winners and our sizer.

Benchmark	Quality Score (lower is better)						Contest Normalized Score (higher is better)					
	1st	2nd	3rd	No Clip.	No Cali.	Ours	1st	2nd	3rd	No Clip.	No Cali.	Ours
NV_NVDLA_partition_m	538.5	471.1	**387.5**	517.9	570.3	482.4	72.0	82.3	**100.0**	74.8	67.9	80.3
NV_NVDLA_partition_p	775.7	753.5	**512.3**	697.4	1113.2	644.8	66.0	68.0	**100.0**	73.5	46.0	79.5
ariane136	2968.5	**1708.6**	1776.8	4362.3	29109.3	2417.1	57.6	**100.0**	96.2	39.2	5.9	70.7
mempool_tile_wrap	1648.2	1358.9	1713.5	3312.5	25768.4	**1356.3**	82.5	100.0	79.3	41.0	5.3	**100.2**
aes_256	8.8	0.2	0.4	36.7	1.9	**0.2**	2.8	100.0	56.0	0.7	12.7	**108.3**
hidden1	15.6	11.5	626.5	16.1	142.4	**0.5**	73.4	100.0	1.8	71.1	8.0	**2569.1**
hidden2	8418.4	12513.7	10737.0	9548.0	31836.0	**8293.9**	100.0	67.3	78.4	88.2	26.4	**101.5**
hidden3	12432.7	28766.2	42833.1	16103.4	16091.7	**11858.4**	100.0	43.2	29.0	77.2	77.3	**104.8**
hidden4	20760.5	69483.3	50919.5	20062.8	44005.7	**15750.4**	100.0	29.9	40.8	103.5	47.2	**131.8**
hidden5	12.4	24.2	234.6	11.3	47.4	**4.5**	100.0	51.3	5.3	109.5	26.2	**274.1**
Average	4757.9	11509.1	10974.1	5466.9	14868.6	**4080.8**	75.4	74.3	58.7	67.9	32.3	**362.0**

TABLE IV: Gate sizer performance comparison based on contest evaluation metrics.

Benchmark	Runtime (s)				TNS (ns)				Slew DRV (ns)				Load DRV (fF)				Power Incre. (uW)			
	1st	2nd	3rd	**Ours**	1st	**2nd**	3rd	Ours	1st	**2nd**	3rd	Ours	1st	2nd	3rd	**Ours**	**1st**	2nd	3rd	Ours
NV_NVDLA_partition_m	11.45	7.65	6.65	**9.55**	-11.80	**-11.80**	-10.23	-11.19	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.85**	1.25	1.02	1.06
NV_NVDLA_partition_p	19.57	27.03	14.49	**15.24**	-23.64	**-20.01**	-17.78	-22.07	0.00	**0.00**	0.09	0.00	0.00	0.00	0.00	**0.00**	**1.04**	1.77	1.10	1.17
ariane136	31.84	111.87	21.91	**24.20**	-73.16	**-21.72**	-27.88	-60.14	12.13	**6.41**	21.28	15.53	0.00	0.00	0.00	**0.00**	**2.18**	6.65	6.06	2.00
mempool_tile_wrap	34.47	81.13	27.79	**27.19**	-0.28	**-0.64**	-2.86	-0.47	30.48	**16.04**	32.72	28.29	1.69	1.23	4.08	**2.18**	**2.52**	8.09	3.99	2.74
aes_256	23.60	66.55	34.89	**20.01**	0.00	**0.00**	0.00	0.00	0.18	**0.00**	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.07**	0.06	0.15	0.10
hidden1	14.06	25.12	8.14	**13.10**	0.00	**0.00**	-2.05	0.00	0.22	**0.10**	3.18	0.00	0.00	0.00	7.73	**0.00**	**0.12**	0.59	0.12	0.13
hidden2	26.12	106.86	22.43	**24.92**	-265.80	**-237.61**	-339.60	-282.20	27.54	**15.34**	27.56	21.20	0.00	0.00	25.92	**0.00**	**2.41**	5.89	4.86	1.80
hidden3	29.15	45.69	25.48	**25.12**	0.00	**0.00**	0.00	0.00	561.16	**557.90**	795.48	548.88	60.44	337.75	1346.13	**44.01**	**0.60**	1.74	0.91	0.61
hidden4	29.82	116.21	32.65	**24.98**	-0.25	**-0.02**	-6.07	0.00	388.27	**342.62**	496.93	366.47	66.49	475.68	555.33	**22.16**	**0.28**	5.87	1.03	1.07
hidden5	24.30	176.03	35.79	**22.30**	-0.50	**0.00**	-10.82	-0.01	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	**0.00**	**2.88**	5.45	2.11	2.98
Ratio	1.18	3.69	1.11	**1.00**	1.00	**0.78**	1.11	1.00	1.04	**0.96**	1.41	1.00	2.63	16.63	39.57	**1.00**	**0.93**	2.64	1.50	1.00

the contestant's gate sizer needs to determine optimal gate sizes for each gate. After gate sizing, another detailed placement and global routing are performed using the open-source tool OpenROAD [44] to ensure a more accurate evaluation.

We follow the contest evaluation metrics, which cover runtime, leakage power, TNS, and severity of slew and load DRV, as well as the formula for calculating quality and normalized scores. The weight of DRV is the highest in this contest, as without DRV elimination the timing report will be not accurate due to extrapolation [39], [40]. Therefore, our strategy emphasizes DRV elimination to guarantee the reliability of subsequent timing analyses and optimization.

The quality score and normalized score are detailed in Table III. These scores highlight the superior capabilities of our sizer. It outperforms the winners in terms of quality score, showcasing its potential for delivering efficient and high-quality gate sizing solutions. For the first three cases, their smaller scale increases the variability in gate sizing optimization results, thus making it difficult to achieve consistently optimal outcomes.

Beyond demonstrating our superiority over the top three winners, we conducted ablation studies to validate the effectiveness of key techniques. Specifically, we examined the impact of removing *gradient clipping* and *gradient calibration*. The results, also presented in Table III, indicate a significant decrease in overall performance when neither technique is excluded, with gradient calibration contributing the most to our model by making the optimization process more targeted and efficient. In particular, performance decline is pronounced in larger designs, highlighting the limited application of the vanilla differentiable approach to large-scale real-world designs.

Table IV presents a detailed comparison of all five contest metrics between the contest winners and our gate sizer, complemented by a radar chart (Figure 5) illustrating the comprehensive and prioritized optimization performance of our gate sizer. In the most critical task of DRV elimination shown on the left side of the radar chart, our gate sizer significantly outperforms others in load DRV reduction while also leading in slew DRV mitigation. For timing, our gate sizer shows competitive results. However, timing metrics may be less reliable here due to the severe DRVs present in half of the benchmarks. Beyond timing and DRV metrics, our gate sizer achieves exceptional power consumption performance, surpassing the second- and third-place winners and approaching the first-place result. Lastly, although all three teams and our gate sizer utilized GPU acceleration or machine learning methods as advocated by the contest, we outperformed the top three winners in runtime, demonstrating remarkable scalability. This efficiency is particularly vital for large-scale VLSI designs, facilitating faster design iterations.

V. Conclusion

In this paper, we highlighted the limitations of applying differentiable frameworks to real-world problems, particularly the challenges of *continuity* and *expressivity*. To address these, we introduced a novel *gradient clipping* strategy to bridge the gap between continuous optimization and the discrete nature of gate sizing, as well as a robust *gradient calibration* framework to ensure precise optimization in complex scenarios. Experimental results demonstrate that our gate sizer achieves exceptional, comprehensive, and well-prioritized

performance, surpassing the top three 2024 ICCAD CAD gate sizing winners in both quality score and runtime. We believe this work establishes a robust foundation for further research and exploration, paving the way for extending the application of differentiable optimization in VLSI design and other practical fields.

REFERENCES

[1] W. C. N. D. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," Patent.

[2] X. Wang *et al.*, "Dasals: Differentiable architecture search-driven approximate logic synthesis," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

[3] Z. Guo and Y. Lin, "Differentiable-timing-driven global placement," in *ACM/IEEE Design Automation Conference (DAC)*, 2022.

[4] G. Chen *et al.*, "Differentiable edge-based opc," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2024.

[5] Y. Du *et al.*, "Fusion of global placement and gate sizing with differentiable optimization," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 2024.

[6] Y.-C. Lu *et al.*, "Lego-size: Llm-enhanced gpu-optimized signoff-accurate differentiable vlsi gate sizing in advanced nodes," in *ACM International Symposium on Physical Design (ISPD)*, 2025.

[7] Y. Ye *et al.*, "Learning-driven physically-aware large-scale circuit gate sizing," 2024. [Online]. Available: https://arxiv.org/abs/2403.08193

[8] R. Liang, A. Agnesina, and H. Ren, "Medpart: A multi-level evolutionary differentiable hypergraph partitioner," in *ACM International Symposium on Physical Design (ISPD)*, 2024.

[9] A. Moola, A. Balu, and A. Krishnamurthy, "Thb-diff: a gpu-accelerated differentiable programming framework for thb-splines," *Engineering with Computers (Eng. Comput.)*, 2023.

[10] T. Xue *et al.*, "Jax-fem: A differentiable gpu-accelerated 3d finite element solver for automatic inverse design and mechanistic data science," *Computer Physics Communications (CPC)*, 2023.

[11] R. Qiu, Z. Sun, and Y. Yang, "Dimes: A differentiable meta solver for combinatorial optimization problems," in *Conference on Neural Information Processing Systems (NIPS)*, 2022.

[12] F. Bu *et al.*, "Tackling prevalent conditions in unsupervised combinatorial optimization: Cardinality, minimum, covering, and more," in *International Conference on Machine Learning (ICML)*, 2024.

[13] B. Adcock and N. Dexter, "The gap between theory and practice in function approximation with deep neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2001.07523

[14] N. Baker *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," 2 2019. [Online]. Available: https://www.osti.gov/biblio/1478744

[15] W. Li *et al.*, "Dgr: Differentiable global router," in *ACM/IEEE Design Automation Conference (DAC)*, 2024.

[16] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations (ICLR)*, 2019.

[17] W. Ning, "Strongly np-hard discrete gate-sizing problems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 13, no. 8, pp. 1045–1051, 1994.

[18] B.-Y. Wu *et al.*, "2024 iccad cad contest problem c: Scalable logic gate sizing using ml techniques and gpu acceleration," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.

[19] Y. Liu and J. Hu, "Gpu-based parallelization for fast circuit optimization," in *ACM/IEEE Design Automation Conference (DAC)*, 2009.

[20] S. Hu, M. Ketkar, and J. Hu, "Gate sizing for cell-library-based designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 6, pp. 818–825, 2009.

[21] Y. Liu and J. Hu, "A new algorithm for simultaneous gate sizing and threshold voltage assignment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2010.

[22] J. P. Fishburn, "Tilos: A posynomial programming approach to transistor sizing," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2003.

[23] J. Hu *et al.*, "Sensitivity-guided metaheuristics for accurate discrete gate sizing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012.

[24] A. B. Kahng *et al.*, "High-performance gate sizing with a signoff timer," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013.

[25] Y.-C. Lu *et al.*, "Rl-sizer: Vlsi gate sizing for timing optimization using deep reinforcement learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2021.

[26] S. Nath *et al.*, "Transsizer: A novel transformer-based fast gate sizer," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[27] C.-K. Cheng *et al.*, "Dagsizer: A directed graph convolutional network approach to discrete gate sizing of vlsi graphs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, no. 4, 2023.

[28] P. Pham and J. Chung, "Agd: A learning-based optimization framework for eda and its application to gate sizing," in *ACM/IEEE Design Automation Conference (DAC)*, 2023.

[29] Y. Du *et al.*, "Powpredict: Cross-stage power prediction with circuit-transformation-aware learning," in *Proceedings of the 61st Annual Design Automation Conference (DAC)*. ACM, 2024.

[30] C.-P. Chen, C. Chu, and D. Wong, "Fast and exact simultaneous gate and wire sizing by lagrangian relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 18, no. 7, pp. 1014–1025, 1999.

[31] S. Daboul *et al.*, "Provably fast and near-optimum gate sizing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 12, pp. 3163–3176, 2018.

[32] C.-P. Chen, C. Chu, and D. Wong, "Fast and exact simultaneous gate and wire sizing by lagrangian relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1014–1025, 1999.

[33] D. Chinnery and A. Sharma, "Integrating lr gate sizing in an industrial place-and-route flow," in *ACM International Symposium on Physical Design (ISPD)*, 2022.

[34] A. Sharma *et al.*, "Fast lagrangian relaxation-based multithreaded gate sizing using simple timing calibrations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 7, pp. 1456–1469, 2020.

[35] M. M. Ozdal, S. Burns, and J. Hu, "Algorithms for gate sizing and device parameter selection for high-performance designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 10, pp. 1558–1571, 2012.

[36] D. Mangiras, D. Chinnery, and G. Dimitrakopoulos, "Task-based parallel programming for gate sizing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 4, pp. 1309–1322, 2023.

[37] X. Zhou *et al.*, "Heterogeneous graph neural network-based imitation learning for gate sizing acceleration," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022.

[38] A. Sharma *et al.*, "Fast lagrangian relaxation based gate sizing using multi-threading," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015.

[39] N. Ohkubo and K. Usami, "Delay modeling and static timing analysis for mtcmos circuits," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2006.

[40] H. Wang, "Modeling and analysis of single-event transient sensitivity of a 65 nm clock tree," *Microelectronics Reliability*, 2018.

[41] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 1, pp. 70–83, 2008.

[42] W. Li *et al.*, "Calibration-based differentiable timing optimization in nonlinear global placement," in *ACM International Symposium on Physical Design (ISPD)*, 2024.

[43] B. Fu *et al.*, "Hybrid modeling and weighting for timing-driven placement with efficient calibration," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024.

[44] The-OpenROAD-Project, "Openroad," GitHub repository, 2024, available online: https://github.com/The-OpenROAD-Project/OpenROAD.