Timing-Driven Global Placement with Hybrid Heuristics and Nadam-based Net Weighting

Linhao Lu[1], Wenxin Yu[1*], Hongwei Tian[1], Chenjin Li[1], Xinmiao Li[1], Zhaoqi Fu[1], Zhengjie Zhao[1], Jingwei Lu[2]
[1]Southwest University of Science and Technology, Mianyang, Sichuan, China
yuwenxin@swust.edu.cn
[2]TikTok
francesco.ljw@gmail.com

***Abstract*—Timing optimization is critical to the entire design flow of the very-large-scale integrated (VLSI) circuit, and Global Placement is pivotal in achieving timing closure within the design flow of very-large-scale integration circuits. However, most global placement algorithms focus on optimizing wirelength rather than timing. Therefore, we propose a novel timing-driven global placement algorithm to address this gap. This paper proposes a timing-driven global placement algorithm utilizing a Nadam-based net-weighting strategy. Additionally, we employ a hybrid heuristic approach for adaptive dynamic adjustment of net weights. The experimental results on the ICCAD 2015 contest benchmarks show that compared to the RePlAce, our algorithm significantly improves WNS and TNS by 40.7% and 56.5%, respectively.**

***Index Terms*—VLSI, Global Placement, STA, Nadam Method, Hybrid Heuristic strategy**

I. Introduction

Global placement is crucial in very-large-scale integration (VLSI) design [1], as it directly affects the timing closure in the VLSI design flow. Moreover, global placement establishes the physical locations of standard cells, thereby exerting an irreversible and significant impact on routing optimization, clock tree construction, and timing signoff verification. As design complexity continues to grow, the intricacy of timing models and the expense of design iterations make achieving timing closure increasingly challenging. Because accurate timing information can only be evaluated at the post-routing stage, commercial design flows often require multiple iterations of core placement and routing to achieve timing closure, thereby slowing down the design process. Circuit placement, which involves determining optimal locations for all circuit components, is a highly challenging task. Therefore, it is essential to consider timing closure during global placement to ensure its successful achievement. However, most current global placement optimization goals and evaluation metrics focus on minimizing total wirelength instead of addressing critical timing paths. In contrast, timing-driven placement is specifically designed to manage the wirelength on these critical paths.

The primary goal of timing-driven placement is to optimize timing metrics such as Total Negative Slack (TNS) and Worst Negative Slack (WNS). Current timing-driven placement techniques can be categorized into net-weighting approaches and path-weighting approaches.

Net-based approaches focus on the design's nets. They use timing information obtained from static timing analysis and feed it back into the placement process through net-weighting or net constraint methods to optimize timing. Static net-weighting approaches, including those based on slack [2] [3] [4] [5] and sensitivity [6] [7], calculate net weights only once during the placement. However, as cells undergo significant changes throughout global placement, early timing estimates are often unreliable, leading to poor performance of static net-weighting. In contrast, dynamic net-weighting [2] updates net weights [8] [9] iteratively throughout the placement process and incorporates net constraint methods [10] [11] [12] to limit the maximum net length.

Path-based approaches [13] [14] [15] [16] directly target specific paths by moving cells to reduce delays in those paths. These approaches are typically formulated as mathematical optimization problems and generally achieve better quality compared to net-based methods. However, as design complexity increases, path-based methods require the analysis of numerous paths, which can be time-consuming. Therefore, path-based approaches are usually applied during the detailed placement stage.

In this paper, we propose a timing-driven global placement algorithm utilizing a Nadam-based [17] net-weighting strategy. We choose the net-weighting approach due to its significant perturbation capabilities and strong scalability. Our primary contributions are as follows:

- **Hybrid Heuristic approach for adaptive dynamic adjustment of net weights.** In timing analysis, a driver with excessive fanout can lead to increased load capacitance. This results in reduced slack or even violations in multiple timing paths connected to that driver. We aim to mitigate the impact of parasitic capacitance and reduce net delay by dynamically adjusting net weights in networks with high fanout, thereby minimizing the adverse effects.
- **Nadam-based net weighting.** In global placement, a net is the smallest unit for wire length calculation. During each timing iteration, we calculate the wire length gradient through the net. Based on the slack of each net, we update the first-order and second-order moments of its gradient,

*Corresponding author.

978-3-9826741-0-0/DATE25

dynamically adjusting the net's weight. The updated net weights are then applied to the Nesterov accelerated gradient descent method in ePlace [18] to help accelerate convergence and optimize timing.
- Experimental results show that on the ICCAD2015 contest benchmark suite [19], after the global placement stage, we achieve an average improvement of 56.5% in Total Negative Slack (TNS) and 40.7% in Worst Negative Slack (WNS) compared to the advanced placer RePlAce [20].

The rest of this paper is structured as follows: section II provides preliminary content, including the basics of nonlinear placement, static timing analysis, and timing optimization. section III introduces the overall flow and detailed explanation of our timing-driven placement algorithm. section III-C presents the experimental results and relevant analysis. Finally, section IV concludes the paper.

II. PRELIMINARIES

In this section, we will introduce related work on nonlinear placement, static timing analysis, and previous timing-driven global placement methods.

A. Nonlinear Global Placement

In the global placement stage, the entire circuit is typically abstracted as a graph. The placer attempts to position millions or even tens of millions of instances in suitable locations to minimize the total wirelength. The netlist $\mathbb{N} = (V, E)$ consists of a set of nets E and a set of nodes (cells) V. Suppose we have n nodes in the design (i.e., $|\mathbb{V}| = n$), global placement seeks positions $(x, y) \in \mathbb{R}^n \times \mathbb{R}^n$ to minimize the total half-perimeter wirelength$W(x, y)$. If we assign a net weight to each net $e \in E$, we can modify the optimization formula to minimize the weighted sum.

$$\min_{x,y} \sum_{e \in E} w_e W(e; \mathbf{x}, \mathbf{y}), \tag{1}$$

where w_e represents the weight of the net $e \in E$.For wirelength smoothing techniques, the log-sum-exp (LSE) model and the weighted-average (WA) model are the most widely used in nonlinear placers to smooth the half-perimeter wirelength (HPWL). We predominantly choose to use the weighted-average (WA) model [21], as shown below.

$$\widetilde{W}_x(e, \gamma; \mathbf{x}, \mathbf{y}) = \frac{\sum_{i \in e} x_i e^{\frac{x_i}{\gamma}}}{\sum_{i \in e} e^{\frac{x_i}{\gamma}}} - \frac{\sum_{i \in e} x_i e^{-\frac{x_i}{\gamma}}}{\sum_{i \in e} e^{-\frac{x_i}{\gamma}}}, \tag{2}$$

$$\widetilde{W}(e, \gamma; \mathbf{x}, \mathbf{y}) = \widetilde{W}_x(e, \gamma; \mathbf{x}, \mathbf{y}) + \widetilde{W}_y(e, \gamma; \mathbf{x}, \mathbf{y}), \tag{3}$$

In Equation2,3, $\widetilde{W}_x(e, \gamma; \mathbf{x}, \mathbf{y})$ and $\widetilde{W}_y(e, \gamma; \mathbf{x}, \mathbf{y})$ represent the net wirelength in the horizontal and vertical directions, respectively, with the hyperparameter γ controlling the approximation accuracy. This equation serves as the wirelength model used in the remainder of this paper.Finally, the nonlinear placement objective is expressed as follows:

$$\min_{x,y} \sum_{e \in E} w_e W(e; \mathbf{x}, \mathbf{y}) + \lambda D(\mathbf{x}, \mathbf{y}), \tag{4}$$

In the Equation4, E represents the set of nets, $W(E; \cdot; \cdot)$ is the wirelength function calculating the total length of a specific net $e \in E$, and $D(\cdot; \cdot)$ denotes the total density penalty function, with λ being the corresponding density weight. This forms a typical unconstrained optimization problem, where the parameters $\mathbf{x}$ and $\mathbf{y}$ are the locations of the cells. It is required that the objective function be differentiable, enabling the use of gradient-based methods to optimize the variables. Additionally, $W(e; \mathbf{x}; \mathbf{y})$ is a nonlinear approximation of the net wirelength.

B. Static Timing Analysis

Static Timing Analysis (STA) is a critical step in evaluating the timing performance of circuits [22]. As illustrated in Figure 1, STA models the circuit as a directed acyclic graph (DAG), where net arcs and cell arcs indicate the direction of signal propagation. As signals traverse these arcs, they introduce delays and voltage changes. By performing minimum and maximum operations on the arrival times of fan-in signals, the accumulated delays determine the earliest and latest arrival times at the pins. These arrival times are used to simulate worst-case timing scenarios and to conduct setup and hold checks [23] [24].

The worst-case performance of a circuit is quantified using setup slack and hold slack, which are calculated based on the arrival times and the required arrival times at the timing endpoints.

$$\text{slack}_{\text{setup}}(p) = \text{rat}_{\text{early}}(p) - \text{at}_{\text{late}}(p), \tag{5}$$

$$\text{slack}_{\text{hold}}(p) = \text{at}_{\text{early}}(p) - \text{rat}_{\text{late}}(p). \tag{6}$$

In Equation6, $\text{slack}_{\text{setup}}(p)$ and $\text{slack}_{\text{hold}}(p)$ represent the setup slack and hold slack at Specific pin p, respectively. Generally, positive slack indicates that the signal meets the timing constraints, while negative slack indicates a timing violation.

In timing optimization tasks, the optimization objective and evaluation metrics always aim to minimize the absolute values of the worst negative slack(WNS) and the total negative slack(TNS) [12] across all timing endpoints. The formula is as follows:

$$\text{WNS}_{\text{setup/hold}} = \min_{\text{endpoint } p'} \text{slack}_{\text{setup/hold}}(p'), \tag{7}$$

$$\text{TNS}_{\text{setup/hold}} = \sum_{\text{endpoint } p'} \min\left(0, \text{slack}_{\text{setup/hold}}(p')\right). \tag{8}$$

C. Timing-Driven Global Placement with Net Weighting

Timing-driven placement focuses on optimizing timing rather than wire length. The timing evaluation metrics are WNS and TNS. Intuitively, WNS represents the worst slack path information, while TNS indicates the overall timing slack of the entire design.

$$\max_{x,y} s(x, y) \text{ s.t. } \rho_b(x, y) \leq \rho_t, \forall b \in B \tag{9}$$

In the equation9, the objective function $s(x, y)$ represents the negative slack function, which can be divided into TNS $s_{\text{tns}}(x, y)$ or WNS $s_{\text{wns}}(x, y)$. Equation (10) shares the same cell

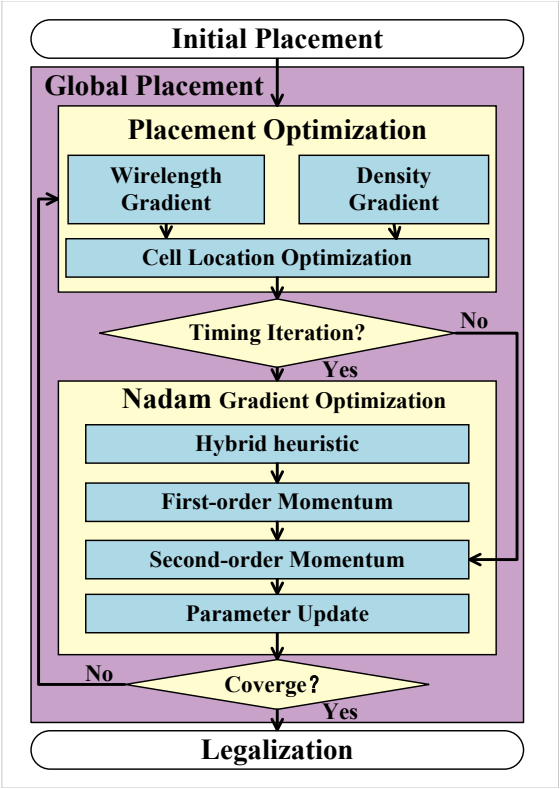

Fig. 1. Our overall flow with timing optimization.

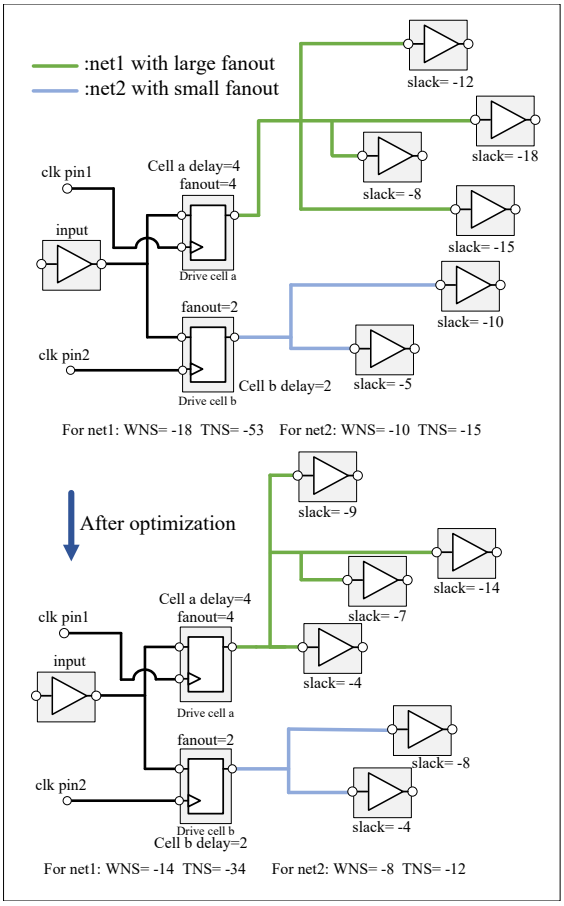

Fig. 2. The impact of drive cell fanout values on network timing

constraints but uses a completely different objective function compared to wirelength-driven analytical placement, which is directly related to cell positions. B is a set of $m \times m$ planar grid cells (bins) for a positive integer m. $\rho_b(x, y)$ denotes the density of a bin $b \in B$, and ρ_t represents the target placement density for each bin.

The net weighting method for timing-driven placement involves calculating the criticality c_e of nets to increase their length weights, thereby improving the Worst Negative Slack (WNS) and Total Negative Slack (TNS) in timing performance. Typically, the criticality c_e of a net is derived by calculating its slack value.

III. ALGORITHMS

In this section, we provide a detailed introduction to our proposed Nadam-based and Hybrid Heuristics net-weighting for timing-driven placement. Our timing-driven global placement flow is illustrated in Figure 1. Unlike classical placement approaches, we incorporate timing analysis and gradient changes to guide the global placement.

A. Static Timing Analysis in Placement

RC Tree Construction: Time-driven placement must be guided by timing analysis, necessitating the reconstruction of an RC tree during each timing analysis iteration due to significant changes in cell positions between consecutive iterations. Each timing iteration provides the current placement solution along with the positions of each cell and pin. Starting from these pin positions, we execute FLUTE [12] to construct the rectilinear minimum Steiner tree for the network. As depicted, these Steiner tree structures reflect the logical layout of our circuit. Before constructing the RC tree, we specify the required unit length resistance r and unit length capacitance c to derive the RC information from the interconnections.

Delay Calculation: In timing analysis, the calculation of delay is illustrated in the figure. When constructing the RC tree, we use the Π-model to input the relevant RC information into the timer and apply the Elmore delay model [25] to approximate the actual delay. After inputting the delay and RC information, we perform static timing analysis on the current placement solution.

B. Hybrid Heuristics of Net timing criticality

In global placement, we assign appropriate weights to each net. This allows us to implicitly convey certain information through the weight values during the global placement phase, As shown in Figure 2. For example, we calculate the criticality of each net to approximately represent its negative slack information.

We select all nets with timing violations (nets with negative slack) and roughly represent their criticality by the ratio of their slack value to the WNS. If a net has positive slack, we consider that it does not require additional weighting, and its criticality is set to zero.

$$c_{\text{slack}_e} = \begin{cases} 0, & \text{if } s_{\text{wns}} \geq 0; \\ \max\left\{0, \frac{s_e}{s_{\text{wns}}}\right\}, & \text{otherwise,} \end{cases} \tag{10}$$

Let c_{slack_e} and s_{wns} represent the criticality of a specific net e and the worst negative slack, respectively. where s_e is the net slack of e.

Flip-flops with excessive fanout increase load capacitance, resulting in longer wirelengths and more complex circuits, which negatively impact timing. However, since the fanout of flip-flops is determined during the logic synthesis stage, we can reduce these negative effects by increasing sensitivity to fanout during the placement stage. The sensitivity parameter for net weighting with respect to fanout is shown in the following formula:

$$c_{\text{fanout}_e} = \tau \frac{\sum_{\text{output pin} \in e} \text{pins}}{\sum_{\text{output pin} \in \text{input pin}} \text{pins}} \tag{11}$$

c_{fanout_e} represents the parameter for the criticality of fanout, τ is a normalization adjustment hyperparameter, and output pin $\in e$ denotes the output pin within the net e. output pin $\in$ input pin refers to all the output pins driven by the output pin in that net. By using Equation 11, a critical value for fanout is obtained, and the positions of the pins are adjusted during placement to mitigate the timing impact caused by high fanout.

The critical value for the current iteration is defined by multiplying c_{slack_e} and c_{fanout_e}, and then normalizing the result.

$$c_e = (c_{\text{slack}_e} \cdot \mu c_{\text{fanout}_e}) \tag{12}$$

μ is a dynamic parameter that balances the critical values of slack and fanout. During the initial stages of placement, the positions of cells undergo significant changes, making slack values unreliable. Therefore, in the early stages, we place more emphasis on fanout. The calculation formula for μ is provided below.

$$\mu = \mathrm{e}^{2 \cdot overflow - 1} \tag{13}$$

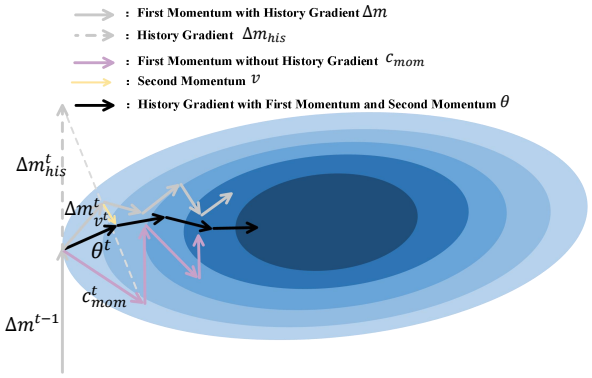

Fig. 3. Schematic diagram of momentum gradient

C. Nadam-based Net Weighting

We introduce a Nadam-based net weighting scheme that leverages both first-order and second-order momentum to dynamically adjust the weighting of individual nets during the timing-driven placement optimization flow. Specifically, for a given net e, we utilize the first-order momentum to update the net's weight at each timing iteration according to the following formula:

$$\tilde{m}_e^{(t+1)} = \tilde{m}_e^{(t)} + \Delta\tilde{m}_e^{(t)}, \quad t \in \mathbb{T}. \tag{14}$$

For different nets, their critical values vary, and after each timing iteration, the critical values for all nets are updated. We dynamically update the net weights during timing iterations using critical values through first-order momentum. In each timing iteration, we calculate the critical values and incorporate them incrementally into the existing momentum to update it, as shown in Equation 14. The specific calculations between momentum and critical values are as follows.

Define the total number of timing iterations as T. For net e, the critical value at iteration t is $c_e^{(m)}$. Define the momentum at the t-th timing iteration as $\hat{m}_e(t)$, and the momentum increment as $\Delta\hat{m}_e(t)$. The formulas are shown as follows:

$$\tilde{m}_e^{(t)} = ln(k \cdot m_e^{(t)}) \tag{15}$$

$$\tilde{c}_e^{(t)} = ln(\omega \cdot c_e^{(t)}) \tag{16}$$

Here, $t \in T$ represents the current iteration number in the timing iteration, and T is the total number of timing iterations. ω and k are two hyperparameters within the range [0, 1], dynamically adjusted based on the overflow in the placement. Their values are inversely related to the overflow value. $\Delta\tilde{m}_e(t)$ represents the net weight and its increment during the t-th timing iteration.

Since we obtain a new momentum in each timing iteration, the increment $\Delta\tilde{m}_e(t)$ is a gradient determined by timing metrics. We aim to emphasize the momentum $\hat{m}_e(t)$ using $c_e(t)$. For $t \in T$, the increment $\Delta\tilde{m}_e(t)$ is given by:

$$\Delta\tilde{m}_e^{(t+1)} = \beta_1 \Delta\tilde{m}_e^{(t)} + (1 - \beta_1)\tilde{c}_e^{(t)}, \quad t \in \mathbb{T}. \tag{17}$$

Equations17 are derived from the first-order momentum method used in gradient descent for training neural networks via backpropagation. The term $\Delta\tilde{m}_e^{(t)}$ can be regarded as velocity, as indicated by Equation14 Generally, the momentum does not align with the gradient of the objective function, so historical gradients are recorded during each timing iteration to guide the actual gradient increment.Each calculation of momentum provides guidance on historical momentum and gradually increases as the placement progresses. The momentum formula is as follows:

$$\Delta\tilde{m}^{(t+1)} = \beta_1 \Delta\tilde{m}^{(t)} + (1 - \beta_1)\tilde{c}^{(t)}, \quad t \in \mathbb{T}. \tag{18}$$

In timing-driven global placement, the cost of timing iterations is significant, leading to considerable resource and

runtime overhead. As a result, the number of timing iterations constitutes only a small portion of the total iterations in the overall timing-driven global placement process.The first-order momentum is updated only during timing iterations. In non-timing iterations, the accuracy of the first-order momentum's direction diminishes as the non-timing iterations count increases. To address this, we introduce the second-order momentum method: calculating second-order momentum during non-timing iterations for fine-tuning the net weights.

$$\tilde{v}^{i+1} = \beta_2 \tilde{v}^i + (1-\beta_2)\tilde{c}_i^2, \quad i \in \mathbb{I}. \tag{19}$$

The second-order momentum v_i tracks the variance of the gradient and enables adaptive parameter allocation for each parameter. Parameters with larger gradients have smaller update steps, while parameters with smaller gradients have larger update steps. This balances different parameters and improves the efficiency of the optimization algorithm. i represents the current non-timing iteration, and I is the total number of non-timing iterations between two successive timing iterations.

The momentum attenuation coefficient β_1 and β_2 are dynamic parameters, and we expect the decrease in momentum in the early stage of layout to have an impact on the later stage of layout. We dynamically adjust it through the overflow value, and the formula is as follows:

$$\beta_1 = e^{0.5-0.4overflow} - 1 \tag{20}$$

$$\beta_2 = \frac{1}{\ln(3overflow + 2.5)} - 0.5 \tag{21}$$

For both the first-order and second-order momentum terms, we apply a correction based on the current decay rates in order to reduce the bias introduced by the momentum estimation.

$$m^t = \frac{\tilde{m}^t}{1-\beta_1^t} \tag{22}$$

$$v^i = \frac{\tilde{v}^i}{1-\beta_2^i} \tag{23}$$

Finally, we update our net weight parameters from the adjusted first-order and second-order momentum, and incorporate these into the net gradient of the objective function to achieve timing-driven placement through net weighting.

$$\theta^{i+1} = \theta^i - \alpha\left(\frac{\beta_1 m^t + (1-\beta_1)\Delta m^t}{\sqrt{v^i}}\right) \tag{24}$$

Here, θ_i represents the net weight in the current iteration, while θ_{i+1} represents the net weight in the next iteration. $\alpha \in [0,1]$ is a hyperparameter used to guide the weights using momentum.The overall momentum gradient diagram is shown in Figure3.

EXPERIMENTAL RESULTS

We conducted experiments on the ICCAD 2015 contest benchmark suite [19]. TableI provides the circuit design parameters of this benchmark suite. Most cases contain over a million cells and nets, making them relatively large, and none of the benchmark cases include movable macros. Our algorithm is implemented in C++ based on the open-source placer RePlAce [20] and the open-source timer OpenSTA [27], utilizing CPU resources for computation. Our testing platform is a 64-bit Linux machine with a 24-core Intel i9-13900k CPU running at 3.0GHz, and 64GB of RAM with a frequency of 4800Hz.

TABLE I
STATISTICS OF THE ICCAD2015 CONTEST BENCHMARKS [19]

case name	#cells	#nets	#pins	#rows
superblue1	1209716	1215710	3767494	1829
superblue3	1213253	1224979	3905321	1840
superblue4	795645	802513	2497940	1840
superblue5	1086888	1100825	3246878	2528
superblue7	1931639	1933945	6372094	3163
superblue10	1876103	1898119	5560506	3437
superblue16	981559	999902	3013268	1788
superblue18	768068	771542	2559143	1788

TableII presents an overall comparison of WNS, TNS, HPWL, and runtime. Our Nadam-based Net Weighting Placer significantly outperforms the state-of-the-art net weighting timing-driven placer, showing substantial improvements on these benchmarks. Notably, on the superblue16 benchmark, we achieved remarkable results with WNS and TNS improvements of 77.7% and 71.2%, respectively, compared to the original RePlAce [20]. Additionally, on the superblue5 benchmark, we improved the convergence of HPWL, reducing the HPWL length by over 64.4% compared to the original RePlAce [20].

As shown in TableIII, we utilize the ICCAD 2015 timing evaluator to assess our timing results. The experimental results demonstrate that our method exhibits advantages in various aspects of net weighting in timing-driven placement. Compared to the state-of-the-art timing-driven placement tool, DreamPlace4.0 [5], we achieved improvements of 14.6% in Worst Negative Slack (WNS), 27.3% in Total Negative Slack (TNS), and 10% in Half-Perimeter Wirelength (HPWL). Although our approach shows a slight disadvantage in WNS compared to the current best-performing timing-driven placement method, Differentiable-Timing-Driven Global Placement(Differentiable TDP) [26], we achieved notable improvements of 1.5% in TNS and 5.8% in HPWL, which are more critical metrics.

IV. CONCLUSION

In this paper, we propose a hybrid heuristic Nadam-based Net Weighting timing-driven placement algorithm. This approach employs multiple hybrid heuristic algorithms to extract the timing-critical values of nets, using the Nadam [17] method to adjust gradients via first-order momentum during timing iterations, and second-order momentum to adjust first-order momentum during non-timing iterations. Additionally, our algorithm demonstrates higher convergence on the superblue5 benchmark, significantly enhancing the convergence of the original placer. These improvements are achieved without excessive runtime overhead, making the quality gains well worth the time investment.

TABLE II

COMPARING WNS, TNS, HPWL AND RUNTIME WITH THE ORIGINAL REPLACE [20] . THE BEST RESULTS ON WNS, TNS AND RUNTIME ARE BOLD-FACED.WNS: IN $(\times 10^3)$PS. TNS: IN $(\times 10^6)$PS. HPWL: HALF-PERIMETER WIRELENGTH $(\times 10^6)$, THE RUNTIME UNIT OF RUNTIME IS SECOND.

Benchmark	RePlAce [20]				Ours			
	WNS	TNS	HPWL	Runtime	WNS	TNS	HPWL	Runtime
superblue1	-142.638	-639.081	391.441	954	**-94.022**	**-322.743**	425.426	1843
superblue3	-203.577	-407.576	447.946	1053	**-181.341**	**-259.943**	465.275	2112
superblue4	-130.881	-372.014	294.423	442.5	**-8.492**	**-82.924**	305.141	923
superblue5	-328.81	-2499.53	1364.813	1130.5	**-241.428**	**-592.289**	**485.38**	2045
superblue7	-185.283	-699.321	533.497	1802.2	**-68.612**	**-285.979**	578.34	3629
superblue10	-194.119	-918.135	868.581	1557.5	**-110.005**	**-521.63**	885.674	3358
superblue16	-153.281	-1101.12	343.303	578.9	**-34.116**	**-316.374**	381.438	1612
superblue18	-92.443	-681.198	214.431	626.5	**-60.204**	**-150.437**	234.093	1350
Avg. Ratio	**2.004**	**2.702**	**1.172**	**0.478**	**1.000**	**1.000**	**1.000**	**1.000**

TABLE III

BASED ON THE RESULTS FROM THE ICCAD 2015 TIMING EVALUATOR, COMPARING WNS, TNS AND HPWL WITH DREAMPLACE4.0 [5] AND DIFFERENTIABLE-TIMING-DRIVEN GLOBAL PLACEMENT [26] . THE BEST RESULTS ON WNS, TNS AND RUNTIME ARE BOLD-FACED.WNS: IN $(\times 10^3)$PS. TNS: IN $(\times 10^5)$PS. HPWL: HALF-PERIMETER WIRELENGTH $(\times 10^6)$.

Benchmark	DreamPlace4.0 [5]			Differentiable TDP [26]			Ours		
	WNS	TNS	HPWL	WNS	TNS	HPWL	WNS	TNS	HPWL
superblue1	-14.103	-85.032	443.1	**-10.770**	-74.854	**423.8**	-15.89	**-61.641**	425.4
superblue3	-16.434	-54.742	482.4	**-12.374**	**-39.430**	478.4	-23.891	-43.758	**465.3**
superblue4	-12.781	-144.380	335.9	-8.492	-82.924	312.2	**-8.33**	**-80.824**	**305.1**
superblue5	-26.760	**-95.782**	556.2	**-25.212**	-108.076	488.7	-37.946	-156.722	**485.4**
superblue7	-15.216	-63.863	604.0	-15.216	-46.426	602.1	**-14.344**	**-36.511**	**578.3**
superblue10	-31.880	-768.748	1036.7	-21.974	-558.054	934.4	**-19.970**	**-477.329**	**885.7**
superblue16	-12.112	-124.181	448.1	-10.854	-87.026	485.1	**-10.527**	**-79.217**	**381.4**
superblue18	-11.871	-47.246	253.6	-7.987	**-19.314**	243.6	**-7.695**	-25.753	**234.1**
Avg. Ratio	**1.146**	**1.273**	**1.1**	**0.889**	**1.015**	**1.058**	**1.000**	**1.000**	**1.000**

V. ACKNOWLEDGMENT

This research is supported by the project of the Ministry of Industry and Information Technology High-Quality Development Program (No. CEIEC-2024-ZM02-0056), the Sichuan Provincial Party Committee's Military-Civilian Integration Committee, and the IEDA Laboratory of Southwest University of Science and Technology.

REFERENCES

[1] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in vlsi placement research," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 275–282, 2012.

[2] M. Burstein and M. N. Youssef, "Timing influenced layout design," in *22nd ACM/IEEE Design Automation Conference*, pp. 124–130, IEEE, 1985.

[3] H. Chang, E. Shragowitz, J. Liu, H. Youssef, B. Lu, and S. Sutanthavibul, "Net criticality revisited: An effective method to improve timing in physical design," in *Proceedings of the 2002 international symposium on Physical design*, pp. 155–160, 2002.

[4] T. Kong, "A novel net weighting algorithm for timing-driven placement," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pp. 172–176, 2002.

[5] P. Liao, D. Guo, Z. Guo, S. Liu, Y. Lin, and B. Yu, "Dreamplace 4.0: Timing-driven placement with momentum-based net weighting and lagrangian-based refinement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 10, pp. 3374–3387, 2023.

[6] B. Halpin, C. R. Chen, and N. Sehgal, "A sensitivity based placer for standard cells," in *Proceedings of the 10th Great Lakes symposium on VLSI*, pp. 193–196, 2000.

[7] T.-Y. Wang, J.-L. Tsai, and C. C.-P. Chen, "Sensitivity guided net weighting for placement driven synthesis," in *Proceedings of the 2004 international symposium on Physical design*, pp. 124–131, 2004.

[8] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proceedings of the 35th annual Design Automation Conference*, pp. 269–274, 1998.

[9] B. Obermeier and F. M. Johannes, "Quadratic placement using an improved timing model," in *Proceedings of the 41st annual Design Automation Conference*, pp. 705–710, 2004.

[10] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI physical design: from graph partitioning to timing closure*, vol. 312. Springer, 2011.

[11] T. Gao, P. M. Vaidya, and C. L. Liu, "A performance driven macro-cell placement algorithm.," in *DAC*, pp. 147–152, 1992.

[12] K. Rajagopal, T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary, and B. Halpin, "Timing driven force directed placement with physical net constraints," in *Proceedings of the 2003 international symposium on Physical design*, pp. 60–66, 2003.

[13] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Parasuram, and B. Halpin, "How accurately can we model timing in a placement engine?," in *Proceedings of the 42nd annual Design Automation Conference*, pp. 801–806, 2005.

[14] M. A. Jackson and E. S. Kuh, "Performance-driven placement of cell based ic's," in *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pp. 370–375, 1989.

[15] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference*, pp. 211–215, 1995.

[16] T. Hamada, C.-K. Cheng, and P. M. Chau, "Prime: A timing-driven placement tool using a piecewise linear resistive network approach," in *Proceedings of the 30th international Design Automation Conference*, pp. 531–536, 1993.

[17] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[18] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "eplace: Electrostatics-based placement using fast fourier transform and nesterov's method," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, pp. 1–34, 2015.

[19] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "Iccad-2015 cad contest in incremental timing-driven placement and benchmark suite," in

2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 921–926, IEEE, 2015.

[20] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2018.

[21] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "Tsv-aware analytical placement for 3d ic designs," in *Proceedings of the 48th Design Automation Conference*, pp. 664–669, 2011.

[22] J. Bhasker and R. Chadha, *Static timing analysis for nanometer designs: A practical approach*. Springer Science & Business Media, 2009.

[23] R. B. Hitchcock, "Timing verification and the timing analysis program," in *Papers on Twenty-five years of electronic design automation*, pp. 446–456, 1988.

[24] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," *Handbook of Algorithms for Physical Design Automation*, pp. 423–446, 2008.

[25] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of applied physics*, vol. 19, no. 1, pp. 55–63, 1948.

[26] Z. Guo and Y. Lin, "Differentiable-timing-driven global placement," in *Proceedings of the 59th Annual Design Automation Conference 2022*, ACM, 2022.

[27] T. O. Project, "Opensta." https://github.com/The-OpenROAD-Project/OpenSTA, 2024. Accessed: 2024-07-06.