

HPPlacer: A High-Precision Slack-Aware Global Placement Engine

Qinggong Shen¹, Chaoli Zhang¹, Haoyang Xu³, Zhiwen Yu^{2,1}, Bin Guo¹,
Yuxuan Zhao³, Bei Yu³, Tsung-Yi Ho³, and Xing Huang^{1*}

¹School of Computer Science, Northwestern Polytechnical University, Xi'an, China

²College of Computer Science and Technology, Harbin Engineering University, Harbin, China

³Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong

Emails: {shenqinggong180828, zclb}@mail.nwpu.edu.cn, hyxu@link.cuhk.edu.hk,
{zhiwenyu, guob, xing.huang}@nwpu.edu.cn, {byu, tyho}@cse.cuhk.edu.hk

Abstract—Timing-driven global placement plays a decisive role in the final performance of very large-scale integration (VLSI) circuits, but is consistently challenged by the trade-off between design accuracy and efficiency. Most existing methods rely on coarse-grained net-weighting strategies. While these approaches are straightforward to implement, they cannot precisely identify and optimize complex timing paths, such as paths with sharing effects or large slack deviations. To overcome this bottleneck, we propose a high-precision slack-aware global placement engine called HPPlacer, which includes the following three key techniques: 1) a local clock buffer-to-flip-flop connection optimization method, 2) a path-level differentiable timing optimization model, and 3) a dynamic adjustment mechanism-based pin-pair weighting strategy. With the proposed method, efficient chip placement with excellent timing behaviors can be generated automatically within a short period of time. The experimental results on multiple benchmark circuits confirm that HPPlacer leads to significant improvements in both timing performance and wirelength compared to state-of-the-art placement tools.

Index Terms—Critical paths, global placement, high precision, slack-aware, timing-driven.

I. INTRODUCTION

Placement is a critical step in VLSI physical design, where the optimal locations of standard cells and macro blocks are determined, with its quality being a decisive factor for the final performance of the chip. Traditional placement design primarily aims for the minimization of chip area, wirelength, and congestion. However, with the continuous scaling of technology nodes and the ever-increasing operating frequency of chips, timing closure has become the most critical challenge in placement design [1]. In conventional flows, many timing violations are uncovered only late in the design cycle, resulting in repeated and costly iterations to achieve sign-off. Consequently, proactively considering timing constraints during the placement stage, known as timing-driven placement (TDP), has become a major research focus in both academia and industry.

On the other hand, incorporating timing awareness into placement design introduces several complex challenges. First, due to the existence of millions of interconnected and coupled timing paths in a circuit, improving the slack of one path may cause violations in multiple previously timing-closed paths. This makes the computation of a globally optimal solution exceptionally difficult. Second, timing optimization objectives often conflict with traditional physical optimization goals such

as wirelength, density, and routability. To optimize the timing of critical paths, TDP algorithms typically cluster cells along the path closely together, but this may induce severe routing congestion, potentially even preventing the completion of subsequent routing tasks. Furthermore, during the timing optimization phase, significant modifications to the chip layout may undermine previously achieved optimizations in power and wirelength, leading to a considerable degradation in the final chip performance. Consequently, TDP algorithms must achieve a delicate balance among optimization accuracy, computational efficiency, and placement quality.

Existing TDP techniques can be broadly classified into two categories: net-based methods and path-based methods. Net-based methods dynamically assign weights to critical nets based on feedback from static timing analysis (STA), thereby guiding the placer to focus on timing-sensitive regions during optimization [2]–[5]. DREAMPlace 4.0 [3], for example, a typical net-based method, achieves stable timing optimization by incorporating historical iteration information into net weight updates. However, a fundamental limitation of such methods lies in their focus on nets rather than paths. This overlooks the signal propagation structures within critical paths, thus inherently capping their optimization capability.

In contrast, path-based methods directly extract and model timing information of critical paths. They formulate path-level objectives as mathematical programming problems and guide critical path optimization via techniques such as path sliding, constraint decomposition, and dynamic weighting [6]–[10]. For example, a recent study [9] proposed a differentiable timing-driven placement framework that leverages the backpropagation mechanism from deep learning to explicitly optimize path-level timing goals. Nevertheless, since this method applies a monotonically increasing global weight to reinforce all path constraints indiscriminately, it sacrifices placement flexibility and may lead to solutions trapped in local optima.

Although existing TDP designs have achieved a certain degree of trade-off among modeling granularity, optimization accuracy, and computational efficiency, they generally rely on macro-level optimization strategies to drive timing closure, lacking a differentiated regulation mechanism for paths with varying criticality. This often leads to either suboptimal utilization of resources or convergence to local optima due to over-

constraining. To address these limitations, we propose a high-precision slack-aware global placement engine called HPPlacer. The major contributions are listed below.

- We propose a hybrid optimization-based method for local clock buffer (LCB)-flip-flop (FF) interconnections, which effectively reduces timing violations and enhances the quality of clock signal distribution through the efficient integration of greedy search and simulated annealing (SA).
- We introduce a path-level differentiable timing optimization model, which optimizes both timing metrics and convergence efficiency through fine-grained path-level timing analysis.
- We develop a pin-pair weight assignment strategy based on a dynamic adjustment mechanism, which adaptively allocates weights according to path slack to further enhance timing performance.
- We validate the proposed method using the ICCAD 2015 standard benchmark [11]. Experimental results demonstrate that HPPlacer achieves significant improvements, with an average reduction of 24.7%, 46.8%, and 9.5% in worst negative slack (WNS), total negative slack (TNS), and half-perimeter wirelength (HPWL), respectively, when compared to DREAMPlace 4.0 [3].

The remainder of this paper is organized as follows. In Section II, the background of TDP and the motivation of this work are described. Section III discusses the proposed HPPlacer in detail. Experimental results are reported in Section IV. Finally, conclusions are drawn in Section V.

II. PRELIMINARY

In this section, we first briefly introduce the two fundamental technologies underlying contemporary TDP, i.e., nonlinear optimization and STA. Then, our motivation is presented by analyzing the drawbacks suffered by existing TDP approaches.

A. Nonlinear Optimization and Static Timing Analysis

The major goal of global placement is to determine the physical locations for millions of logic cells on a chip, which is inherently an NP-hard combinatorial optimization problem [12]. To tackle this complexity, the mainstream approaches transform this problem into a continuous and differentiable nonlinear optimization formulation. The key lies in constructing an objective function $\Phi(x, y)$ that comprehensively reflects placement quality. This function typically consists of two core components: a total wirelength evaluation term that captures circuit connectivity cost, and a density penalty term that prevents cell overlap, defined as

$$\min_{x, y} \Phi(x, y) = \min_{x, y} \left(\sum_{e \in E} w_e W_e(x, y) + \lambda P(x, y) \right), \quad (1)$$

where (x, y) represents the coordinates of a cell, E is the set of nets, w_e is the weight assigned to net $e \in E$, $W_e(x, y)$ is used to estimate the HPWL of e , $P(x, y)$ is the density penalty, and parameter λ is a weighting factor. Note that to make the objective function solvable, smooth approximations are usually applied to the computation of HPWL, and commonly used

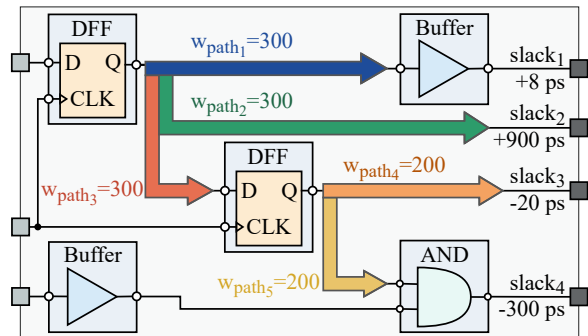


Fig. 1: Analysis of drawbacks suffered by the existing coarse-grained weighting models.

methods include Log-Sum-Exp (LSE) and weighted-average functions.

On the other hand, STA serves as a critical tool for evaluating timing convergence in placement. STA models the circuit as a directed acyclic graph with delay annotations, where nodes and edges represent timing pins and timing arcs indicating signal propagation directions, respectively. Within the STA framework, the key variables are the actual arrival time $A(p)$ and the required arrival time $R(p)$ for each pin. The timing slack of a pin can thus be defined as

$$S_p = R(p) - A(p), \quad (2)$$

where S_p represents the slack of pin p . A negative slack ($S_p < 0$) indicates a timing violation on the corresponding path.

To quantify the satisfaction of timing constraints from a global perspective, two widely adopted timing analysis metrics WNS and TNS are defined as

$$\text{WNS} = \min_{p \in \mathcal{T}} S_p, \quad (3)$$

$$\text{TNS} = \sum_{p \in \mathcal{T}} \min(S_p, 0), \quad (4)$$

where \mathcal{T} is the set of all timing nodes in a circuit.

B. Motivation

As discussed previously, a number of TDP methods, e.g., [2]–[5], have been proposed to incorporate timing optimization into placement design. Most of these methods, however, employ coarse-grained weighting models, resulting in insufficient accuracy in the optimization of complex critical paths. The inherent limitations are analyzed as follows:

(1) **Inability to perform differentiated weight assignments for paths with opposing timing slacks within the same net.** Let us take the circuit shown in Fig. 1 as an example, where path₁ and path₃ originate from the same fan-out node, yet exhibit timing slacks of +900 ps and -20 ps, respectively. Coarse-grained modeling approaches, e.g., DREAMPlace 4.0 [3], assign a uniform weight (300) to all paths based solely on the worst timing path on the net. As a consequence, path₁ with a positive slack is assigned an identical weight as the timing-violation path₃, leading to an over-allocation of optimization resources to a non-critical path.

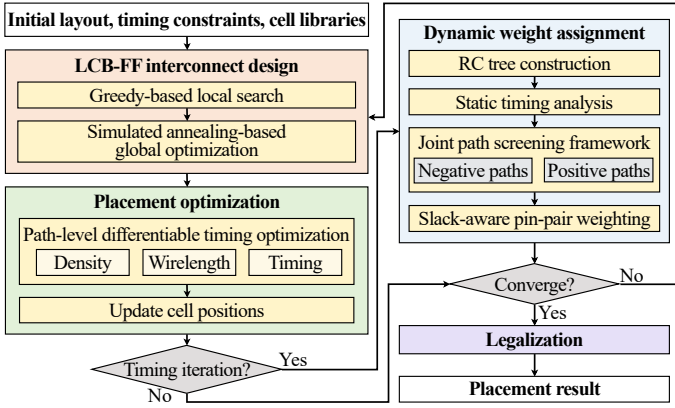


Fig. 2: The overall flow of proposed HPPlacer

(2) **Insensitivity to the magnitude of positive timing slacks.** Let us revisit the circuit shown in Fig. 1. Although the slacks of both $path_1$ and $path_2$ are positive (+900 ps and +8 ps), the timing of the latter is evidently more critical. In coarse-grained weighting models, dominated by the negative slack of the critical path $_3$, both $path_1$ and $path_2$ are assigned an identical weight (300). When the optimizer adjusts shared cells to improve the slack of $path_3$, this may cause $path_2$, which originally had minimal positive slack, to deteriorate into a timing-violation path. Furthermore, assigning an excessively high weight to $path_1$, which has ample slack, reduces placement flexibility and consequently suppresses the optimization potential for other critical paths.

(3) **Failure to identify the convergence effect of multiple critical paths.** In Fig. 1, $path_4$ and $path_5$ are two paths that converge at a common fan-out node and both exhibit timing violations (-20 ps and -300 ps, respectively). This indicates that their upstream shared path acts as a common bottleneck for these two critical paths, demonstrating a significant convergence effect. However, the coarse-grained weighting strategies typically consider only the single worst-case path among them, failing to apply an aggregated weight to the upstream shared path. Consequently, this underestimates the timing urgency of the upstream path, thereby impairing the optimization capability within complex interconnect networks.

Based on the above analysis, it can be concluded that existing TDP methods, which rely on coarse-grained optimization strategies, encounter significant bottlenecks in achieving timing optimization within complex networks. Thus, it is imperative to refine the granularity of optimization and develop a placement methodology that delivers high-precision timing optimization.

III. DETAILS OF PROPOSED METHOD

In this section, we discuss the proposed HPPlacer in detail, which includes three major steps as illustrated in Fig. 2: 1) LCB-FF interconnect design, 2) placement optimization, and 3) dynamic weight assignment.

A. LCB-FF Interconnect Design

To enhance the quality of signal distribution from the source, we first introduce the concept of a virtual clock network. As shown in Fig. 3, by introducing LCBs, the clock tree structure

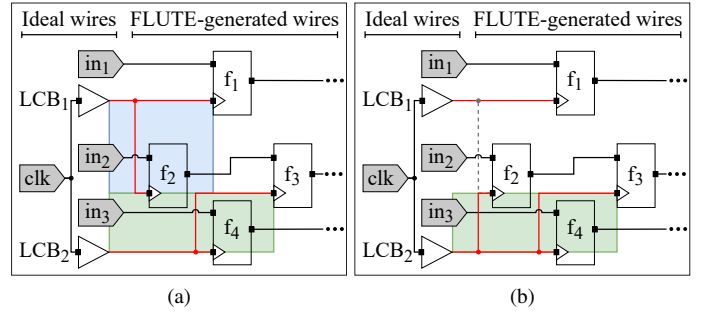


Fig. 3: Greedy optimization of the virtual clock network. (a) LCB-FF connections before optimization, with a total HPWL of 21 and (b) LCB-FF connections after greedy optimization, with a total HPWL of 16.

is abstracted into an LCB-FF connection network, where each LCB is used to drive a set of nearby FFs. Note that modern high-performance clock synthesis techniques can ensure high-precision synchronization among the outputs of different LCBs. Based on this virtual clock network, we propose a hybrid local-global optimization method for the LCB-FF interconnect design. This method aims to reduce the signal arrival delay at FFs, thereby increasing the overall timing slack. The problem can be formulated as follows: Given a set $\mathcal{F} = \{f_i | 1 \leq i \leq |\mathcal{F}|\}$ of FFs with their physical locations, a set $\mathcal{B} = \{b_i | 1 \leq i \leq |\mathcal{B}|\}$ of LCBs with their physical locations, and a maximum load capacitance C_{max} associated with each LCB, find a mapping $\mathcal{A} : \mathcal{F} \rightarrow \mathcal{B}$ such that the total HPWL of all LCB-FF connections is minimized. The optimization objective can be defined as

$$\begin{aligned} \min_{\mathcal{A}} \sum_{f_i \in \mathcal{F}} \text{HPWL}(f_i, \mathcal{A}(f_i)) \\ \text{subject to } \forall l_j \in \mathcal{B}, \text{Load}(b_j) \leq C_{max}. \end{aligned} \quad (5)$$

To solve this problem, we propose a hybrid optimization algorithm that integrates greedy search with SA, which are utilized for local tuning and global optimization, respectively. To accelerate the solving process, we introduce an adaptive spatial neighborhood search strategy based on the statistical features of HPWL. This strategy dynamically adjusts the optimal search range for cells according to the bounding box dimensions of all LCB-FF nets in the initial placement. Specifically, we first represent the position of a cell k as a vector $\vec{p}_k = (x_k, y_k)$. For each LCB b_j and its fan-out set of FFs, denoted as \mathcal{C}_j , under an initial mapping \mathcal{A}_0 , we compute its dimension vector \vec{S}_j as

$$\vec{S}_j = \begin{pmatrix} W(\mathcal{C}_j) \\ H(\mathcal{C}_j) \end{pmatrix} = \begin{pmatrix} \max_{k \in \mathcal{C}_j \cup \{l_j\}} \vec{p}_k - \min_{k \in \mathcal{C}_j \cup \{l_j\}} \vec{p}_k \end{pmatrix}, \quad (6)$$

where $W(\mathcal{C}_j)$ and $H(\mathcal{C}_j)$ represent the width and height of the bounding box of \mathcal{C}_j , respectively. Then, a statistical analysis is performed on the dimension vectors of all LCB-FF nets to compute the mean vector $\vec{\mu}_S$ and the standard deviation vector $\vec{\sigma}_S$:

$$\vec{\mu}_S(\mu_W, \mu_H) = \frac{1}{|\mathcal{B}|} \sum_{l_j \in \mathcal{B}} \vec{S}_j, \quad (7)$$

$$\vec{\sigma}_S(\sigma_W, \sigma_H) = \sqrt{\frac{1}{|\mathcal{B}|} \sum_{l_j \in \mathcal{B}} (\vec{S}_j - \vec{\mu}_S)^2}. \quad (8)$$

Consequently, an adaptive search radius R that accommodates different placement scales and congestion conditions can be constructed as

$$R_{search} = \langle \vec{1}, \vec{\mu}_S \rangle + \gamma_{rad} \langle \vec{1}, \vec{\sigma}_S \rangle, \quad (9)$$

where γ_{rad} is a parameter that controls the sensitivity of the neighborhood size to the placement discreteness. This allows the search range to better adapt to the morphological characteristics of the current layout, thereby achieving a balance between algorithmic efficiency and solution quality.

In each iteration, we randomly sample a subset of FFs from \mathcal{F} to form a candidate set \mathcal{F}_{sam} . For each $f_i \in \mathcal{F}_{sam}$, we enumerate the non-overloaded LCBs within its neighborhood and select a new mapping scheme \mathcal{A}' that minimizes the objective in (5). Let h_{orig} and h' be the total HPWL corresponding to the original mapping \mathcal{A} and the updated mapping \mathcal{A}' , respectively. We define the difference, $\Delta = h_{orig} - h'$, as the optimization gain yielded by this update. Therefore, among all feasible update schemes, the algorithm selects the one that maximizes the gain Δ .

Fig. 3 shows an example of this connection remapping. Fig. 3(a) is the LCB-FF interconnect network before optimization, where LCB_1 and LCB_2 are connected to $\{f_1, f_2\}$ and $\{f_3, f_4\}$, respectively. Their corresponding net bounding boxes are depicted by the blue and green rectangles, and the initial total HPWL is 21. After a round of greedy optimization, the driving source for f_2 is remapped from LCB_1 to LCB_2 , as shown in Fig. 3(b). This modification leads to a significant reduction in the bounding box of the net connected to LCB_1 , thus decreasing the total HPWL to 16. This demonstrates that the local adjustment can effectively improve placement quality while satisfying all constraints.

On the other hand, we employ an SA-based mechanism to prevent the search process from being trapped in local optima. This mechanism facilitates a global search by adopting a probabilistic acceptance criterion. The initial temperature T_0 is configured as

$$T_0 = \theta \cdot \sum_{f_i \in \mathcal{F}} (|(\vec{p}_{f_i})_x - (\vec{p}_{A_0(f_i)})_x| + |(\vec{p}_{f_i})_y - (\vec{p}_{A_0(f_i)})_y|), \quad (10)$$

where θ is a coefficient governing the initial temperature, and \vec{p}_{f_i} denotes the coordinate vector of a flip-flop f_i . Throughout the iterative process, the temperature T_k is progressively reduced according to the following cooling schedule:

$$T_{k+1} = \alpha \cdot T_k, \quad (11)$$

where α is the cooling factor and T_k represents the temperature after the k -th iteration.

In each iteration of the SA, an FF is randomly selected from \mathcal{F} to perform either a neighborhood move with probability ϵ or a global perturbation with probability $1 - \epsilon$, thereby ensuring a thorough exploration of the solution space. If the target LCB is not overloaded, the gain Δ resulting from this move is

calculated. When the move yields a positive gain, i.e., $\Delta \geq 0$, it is always accepted. Conversely, if the move leads to a worse solution, i.e., $\Delta < 0$, indicating that the HPWL deteriorates, it is still accepted with a probability of $\exp(-\Delta/T)$. Based on this probabilistic acceptance mechanism, the proposed method effectively avoids becoming trapped in local optima.

B. Placement Optimization

To achieve high-precision timing control, we refine the optimization granularity from the net level to the more fine-grained path level. We first define the concept of a *pin-pair*: On any given timing path, two consecutive pins are considered a pin-pair if they belong to the same net and constitute a direct signal propagation link. Note that a pin-pair can be part of multiple timing paths, and its timing criticality may vary accordingly. Then, the pin-pair is adopted as the fundamental unit, representing the minimum granularity for our optimization. Under the RC delay model, wire resistance R and capacitance C are both proportional to the wire length d . Consequently, the timing delay τ exhibits a quadratic relationship with the length:

$$\tau = R_{wire} \cdot C_{wire} = (k_R \cdot d) \cdot (k_C \cdot d) = k \cdot d^2, \quad (12)$$

where $k = k_R k_C$ is a constant. To accurately model this quadratic behavior during placement, we formulate the timing metric as the square of the equivalent pin-to-pin distance. Furthermore, since the traditional HPWL metric is non-differentiable, we employ the LSE function to create a smooth approximation, thereby ensuring the differentiability of our objective function:

$$\text{LSE}_\gamma(x_i - x_j) = \gamma \ln(e^{(x_i - x_j)/\gamma} + e^{(x_j - x_i)/\gamma}), \quad (13)$$

$$\text{LSE}_\gamma(y_i - y_j) = \gamma \ln(e^{(y_i - y_j)/\gamma} + e^{(y_j - y_i)/\gamma}). \quad (14)$$

The timing metric $\tau(i, j)$ for a pin pair can then be formulated by squaring the sum of the LSE-approximated distances in the horizontal and vertical directions:

$$\tau(i, j) = [\text{LSE}_\gamma(x_i - x_j) + \text{LSE}_\gamma(y_i - y_j)]^2. \quad (15)$$

In our global placement objective function, we incorporate a timing loss term for all pin-pairs on critical paths to precisely guide the optimization. The objective of HPPlacer is formulated as

$$\min_{x, y} \sum_{(p_i, p_j) \in \mathcal{P}_{crit}} w_{ij} \tau(i, j) + \sum_{e \in E} w_e L_e(x, y) + \lambda P(x, y), \quad (16)$$

where the first term represents the path-level timing cost, which aims to enhance circuit performance by minimizing the delay of all pin-pairs within the set of critical paths \mathcal{P}_{crit} . The second term is the global wirelength cost, intended to reduce the total HPWL of all nets. The third term, $\lambda P(x, y)$, is a density penalty that ensures a valid placement by preventing cell overlaps. Moreover, $\tau(i, j)$ and $L_e(x, y)$ are the differentiable estimators for timing and wirelength, respectively, designed for gradient-based optimization. The weights w_{ij} and w_e are dynamically adjusted to prioritize different optimization targets, while λ is a coefficient that balances the penalty strength.

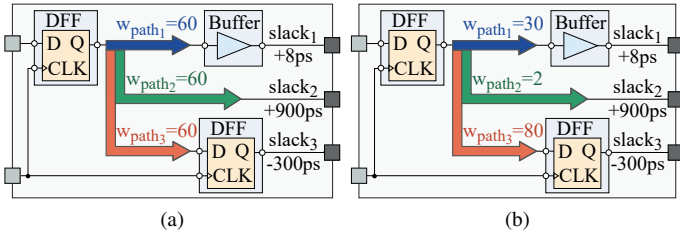


Fig. 4: Comparison of weighting mechanisms. (a) Result generated by the conventional uniform weighting methods and (b) result generated by the proposed weighting method.

To solve the non-linear optimization problem defined in (16), we employ a gradient-based method to update cell locations. In each iteration, the gradient of the objective function $\mathcal{L}(\mathbf{p})$ with respect to each cell’s coordinates is computed as a linear combination of the gradients from the three constituent terms:

$$\nabla \mathcal{L}(\mathbf{p}) = \nabla \mathcal{L}_{\text{time}} + \nabla \mathcal{L}_{\text{wire}} + \lambda \nabla \mathcal{L}_{\text{pen}}, \quad (17)$$

where $\nabla \mathcal{L}_{\text{time}}$, $\nabla \mathcal{L}_{\text{wire}}$, and $\nabla \mathcal{L}_{\text{pen}}$ are the gradients of the timing, wirelength, and penalty terms, respectively.

We then adopt the Nesterov accelerated gradient (NAG) algorithm to update the positions of all cells [13]. Let \mathbf{p}_t denote the coordinate vector at iteration t , \mathbf{v}_t the momentum vector, γ_{mom} the momentum coefficient, and η the learning rate. Instead of moving directly along the current gradient $\nabla \mathcal{L}(\mathbf{p}_t)$, NAG leverages a “lookahead” momentum mechanism. Specifically, the algorithm first performs a preliminary jump to a lookahead position $\mathbf{p}_{\text{lookahead}} = \mathbf{p}_t - \gamma_{\text{mom}} \mathbf{v}_{t-1}$ based on the previous momentum. It then computes the gradient at this lookahead position, $\nabla \mathcal{L}(\mathbf{p}_{\text{lookahead}})$, and uses this gradient to correct the momentum vector. The complete update process can be formulated as

$$\mathbf{v}_t = \gamma_{\text{mom}} \mathbf{v}_{t-1} + \eta \nabla \mathcal{L}(\mathbf{p}_t - \gamma_{\text{mom}} \mathbf{v}_{t-1}), \quad (18)$$

$$\mathbf{p}_{t+1} = \mathbf{p}_t - \mathbf{v}_t. \quad (19)$$

By using the NAG algorithm, HPPlacer can effectively suppress the oscillation problems presented in traditional momentum methods, thereby converging to a high-quality solution with better speed and stability.

C. Dynamic Weight Assignment

To further improve the optimization performance of timing-driven placement, we propose a slack-aware dynamic weight adjustment mechanism. During each placement iteration, the weights of pin-pairs are dynamically adjusted based on path slack. Specifically, weights for pin-pairs on negative-slack paths are strengthened to expedite the resolution of timing violations, while those on positive-slack paths are relaxed to prevent over-optimization. This strategy strikes a harmonious balance between critical path optimization and overall placement quality.

To precisely adjust weights of pin-pairs, we propose a unified framework based on OpenTimer [14] for identifying both positive and negative slack paths. In this framework, two commands, `report_timing(n)` and `report_best_timing(n)`, are employed to extract the n paths with the minimum and

TABLE I: Details of benchmarks used in the experiments

Benchmark	Std. Cells	Macros	Nets	Rows	Area (μm^2)
sblue1	1209284	432	1215710	1829	8543×3131
sblue3	1212678	575	1224979	1840	9174×3150
sblue4	795339	306	802513	1840	5668×3150
sblue5	1086104	784	1100825	2528	9403×4326
sblue7	1931220	419	1933945	3163	6051×5412
sblue10	1874460	1643	1898119	3437	8273×5881
sblue16	981458	101	999902	1788	5642×3061
sblue18	767861	207	771542	1788	4618×3061

maximum slack, respectively. Specifically, for each path ψ , we first compute its slack mapping factor u_ψ as

$$u_\psi = \tanh(\beta \cdot s_\psi), \quad (20)$$

where s_ψ is the slack of path ψ and β is a sensitivity hyperparameter for the mapping function.

Afterward, all mapping factors within the set of paths \mathcal{P}_{ij} that contain the pin-pair (p_i, p_j) are aggregated to derive the total adjustment value:

$$U_{(i,j)} = \sum_{\psi \in \mathcal{P}_{ij}} u_\psi. \quad (21)$$

Finally, the updated weight is computed as

$$w_{\text{new}}(i, j) = w_{\text{base}}(i, j) - U_{(i,j)}, \quad (22)$$

where $w_{\text{base}}(i, j)$ is the initial weight of the pin-pair (p_i, p_j) .

With the proposed pin-pair weighting mechanism, differentiated weights can be applied precisely based on the magnitude of timing slacks. Moreover, convergence effects can also be identified to strengthen constraints on upstream shared paths, thereby overcoming the drawbacks suffered by existing methods. As shown in Fig. 4, compared with the conventional uniform weighting method, differentiated weights are precisely assigned to the three paths based on their timing slacks.

IV. EXPERIMENTAL RESULTS

The proposed HPPlacer was implemented in C++ and tested on a workstation with 3.40-GHz CPU and 64-GB memory. The eight benchmark circuits released by the ICCAD 2015 contest were adopted to evaluate the proposed method as shown in Table I [11], where columns 2–5 are the numbers of standard cells, macro modules, routing nets, and rows in the given circuits, respectively. The last column shows the area of each circuit after floorplanning. It can be seen that all the test cases are relatively large and most of them contain millions of cells and nets. In addition, the parameters used in HPPlacer are set as follows: $\alpha = 0.95$, $\theta = 0.05$, $\epsilon = 0.9$, $\eta = 0.01$, $\lambda = 8 \times 10^{-5}$, $\gamma = 4.0$, $\gamma_{\text{mom}} = 0.9$, $\beta = 0.005$.

To validate the effectiveness of the proposed method, we compare HPPlacer with three recently proposed TDP methods, including DREAMPlace 4.0 [3], Differentiable-TDP [9], and Nadam-NW [15]. The comparison results are reported in Table II. It can be seen that HPPlacer achieves average improvements of 24.8%, 3.3%, and 20.3% in the WNS metric compared to the other three methods, respectively; Regarding the TNS

TABLE II: Comparison results with DREAMPlace 4.0 [3], Differentiable-TDP [9], and Nadam-NW [15]¹

Bench	DREAMPlace 4.0 [3]				Differentiable-TDP [9]				Nadam-NW [15]				HPPlacer			
	WNS	TNS	HPWL	Runtime	WNS	TNS	HPWL	Runtime	WNS	TNS	HPWL	Runtime	WNS	TNS	HPWL	Runtime
sblue1	-14.10	-85.03	443.1	457.3	-10.77	-74.85	423.8	260.4	-15.89	-61.64	425.4	2692.6	-9.94	-29.94	419.9	550.9
sblue3	-16.43	-54.74	482.4	581.4	-12.37	-39.43	478.4	343.6	-23.89	-43.76	465.3	3513.5	-11.84	-30.05	463.3	606.4
sblue4	-12.78	-144.38	335.9	352.7	-8.49	-82.92	312.2	194.4	-8.33	-80.82	305.1	1433.8	-8.36	-87.02	311.1	1114.3
sblue5	-26.76	-95.78	556.2	581.0	-25.21	-108.08	488.7	194.8	-37.95	-156.72	485.4	3351.6	-24.94	-69.67	484.9	700.4
sblue7	-15.22	-63.86	604.0	669.8	-15.22	-46.43	602.1	390.1	-14.34	-36.51	578.3	4573.7	-15.17	-45.17	565.4	1441.7
sblue10	-31.88	-768.75	1036.7	1140.1	-21.97	-558.05	934.4	620.3	-19.97	-477.33	885.7	5500.5	-22.39	-578.86	872.1	2187.9
sblue16	-12.11	-124.18	448.1	457.3	-10.85	-87.03	485.1	296.8	-10.53	-79.22	381.4	2628.9	-10.41	-69.98	400.8	472.2
sblue18	-11.87	-47.25	253.6	255.4	-7.99	-19.31	243.6	230.7	-7.70	-25.75	234.1	2260.6	-7.31	-16.55	221.6	338.0
Avg	1.328	1.880	1.105	0.706	1.034	1.340	1.062	0.420	1.255	1.377	1.006	4.337	1.000	1.000	1.000	1.000

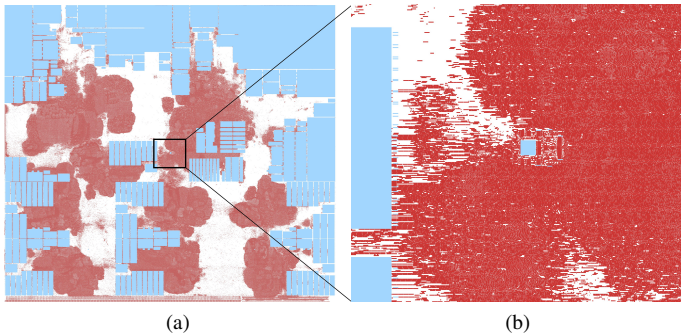


Fig. 5: Final placement results of the largest benchmark sblue7. (a) The overall placement result and (b) enlarged detail of the region marked by the black box in (a).

metric, the advantages of the proposed method are even more pronounced, and the average improvement reaches out 46.8%, 25.4% and 27.4%, respectively. Furthermore, among these methods, HPPlacer also realizes the shortest average HPWL, resulting in the minimization of chip cost. In terms of runtime², HPPlacer exhibits significantly faster performance than Nadam-NW across all the benchmarks, while being marginally slower than DREAMPlace 4.0 and Differentiable-TDP on average. This trade-off is considered acceptable given its substantial improvements in key timing metrics and wirelength.

Based on the aforementioned experimental results, it can be seen that the proposed HPPlacer achieves comprehensive performance superiority compared to the other three placers. We attribute this outcome to three main factors: 1) The incorporation of LCB-FF interconnect design and optimization enables HPPlacer to pre-optimize clock signal arrival time for each flip-flop, which plays a critical role in improving WNS — an aspect neglected by other methods, 2) the proposed path-level differentiable timing optimization model further enhances the accuracy of net weighting, allowing optimization resources to be precisely focused on timing violations. In contrast, coarse-grained methods such as DREAMPlace 4.0 and Nadam-NW apply uniform weights only at the net level, and 3)

¹The units of WNS and TNS are 10^3 ps and 10^5 ps, respectively; The HPWL values are scaled by a factor of 10^6 .

²To ensure a fair comparison, we run HPPlacer and DREAMPlace 4.0 on the same machine. Additionally, by employing the compensation technique from [9], runtimes of Differentiable-TDP and Nadam-NW are normalized to eliminate machine-induced variations.

the dynamic pin-pair weighting strategy effectively strengthens/weakens negative/positive slack paths through bidirectional regulation. This mechanism not only concentrates efforts on repairing critical paths but also avoids over-constraining non-critical paths, thereby preserving greater optimization flexibility for the placer. Consequently, HPPlacer achieves comprehensive optimization of both WNS and TNS without compromising total wirelength (HPWL).

Finally, Fig. 5 shows the placement result generated for benchmark sblue7, which is the largest test case in our experimental evaluation. To minimize the delays of critical paths, HPPlacer proactively clusters associated logic cells, forming several high-density core regions around macro blocks. Furthermore, the overall layout strictly maintains overlap-free constraints, demonstrating that our method effectively manages complex physical constraints while aggressively optimizing timing. Through these analyses, it can be concluded that the proposed HPPlacer effectively coordinates timing optimization with placement quality, providing a highly competitive solution for TDP in VLSI circuits.

V. CONCLUSION

We have investigated the timing-driven placement problem in VLSI design and proposed a high-precision slack-aware method called HPPlacer. The proposed method enhances timing convergence efficiency by optimizing the interconnection relationships between LCBs and FFs, constructing a path-based timing optimization model, and incorporating a dynamic weight update strategy, so that timing violations on critical paths can be mitigated precisely while maintaining overall placement quality. Experimental results on multiple benchmark circuits demonstrate that HPPlacer achieves superior performance in key metrics including WNS, TNS, and HPWL compared to several state-of-the-art timing-driven placers, validating the effectiveness of the proposed methodology.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62572396, the National Key Research and Development Program of China under Grant 2024YFB4505502, the National Natural Science Fund of China for Excellent Young Scientists Fund Program (Overseas), and the Natural Science Foundation of Fujian Province under Grant 2024J01984. Xing Huang is the corresponding author.

REFERENCES

- [1] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in VLSI placement research," in *Proc. of ICCAD*, 2012, pp. 275–282.
- [2] M. Wei, X. Tong, Z. Cai, P. Zou, Z. Lin, and J. Chen, "An analytical placement algorithm with routing topology optimization," in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2024, pp. 294–299.
- [3] 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 TCAD*, vol. 42, no. 10, pp. 3374–3387, 2023.
- [4] D. Mangiras, A. Stefanidis, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering," *IEEE TCAD*, vol. 39, no. 10, pp. 2835–2848, 2019.
- [5] G. Wu and C. Chu, "Two approaches for timing-driven placement by Lagrangian relaxation," *IEEE TCAD*, vol. 36, no. 12, pp. 2093–2105, 2017.
- [6] Y. Du, Z. Guo, Y. Lin, R. Wang, and R. Huang, "Fusion of global placement and gate sizing with differentiable optimization," in *Proc. of ICCAD*, 2024, pp. 1–9.
- [7] Z. Lin, M. Wei, Y. Chen, P. Zou, J. Chen, and Y.-W. Chang, "Electrostatics-based analytical global placement for timing optimization," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.
- [8] W. Li, Y. Kukimoto, G. Servel, I. Bustany, and M. E. Dehkordi, "Calibration-based differentiable timing optimization in non-linear global placement," in *Proc. Int. Symp. Phys. Des.*, 2024, pp. 31–39.
- [9] Z. Guo and Y. Lin, "Differentiable-timing-driven global placement," in *Proc. of DAC*, 2022, pp. 649–654.
- [10] S. N. Shahsavani and M. Pedram, "TDP-ADMM: A timing driven placement approach for superconductive electronic circuits using alternating direction method of multipliers," in *Proc. of DAC*, 2020, pp. 1–6.
- [11] M. C. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 cad contest in incremental timing-driven placement and benchmark suite," in *Proc. of ICCAD*, 2015, pp. 921–926.
- [12] W. E. Donath, "Complexity theory and design automation," in *Proc. of DAC*, vol. 19, Minneapolis, MN, USA, 1980, pp. 412–419.
- [13] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE TCAD*, vol. 38, no. 9, pp. 1717–1730, 2019.
- [14] T. W. Huang and M. D. F. Wong, "Opentimer: A high-performance timing analysis tool," in *Proc. of ICCAD*, Austin, TX, USA, 2015, pp. 895–902.
- [15] L. Lu, W. Yu, H. Tian, C. Li, X. Li, Z. Fu, Z. Zhao, and J. Lu, "Timing-driven global placement with hybrid heuristics and nadam-based net weighting," in *Proc. Design, Autom., and Test Europe Conf.*, 2025, pp. 1–7.