

Timing-Driven Detailed Placement with Collaborative Topology Reconstruction

Zhengjie Zhao^{1†}, Wenxin Yu^{1†*}, Jie Ma¹, Mengshi Gong¹, Youzhi Zheng¹, Xinmiao Li¹, Wenyu Liu¹, Jingwei Lu²

¹Southwest University of Science and Technology, Mianyang, Sichuan, China

Email: yuwenxin@swust.edu.cn

²TikTok

Email: francesco.ljw@gmail.com

Abstract—Placement is a critical step in the physical design, as it largely determines the potential for subsequent optimization. In this work, we propose a timing-driven detailed placement framework: first, a simplified RC-tree model is employed for flip-flop–buffer compensation; then, a gradient-augmented global heuristic algorithm is incorporated; and finally, timing improvement is achieved through local collaborative optimization. A comprehensive evaluation on eight ICCAD 2015 benchmarks demonstrates the effectiveness of our approach. Compared to DREAMPlace4.0-DP, a state-of-the-art timing-driven placer, our framework achieves an average improvement of 19.60% in WNS and 55.74% in TNS, while introducing less disturbance to the global placement. Moreover, it delivers a 0.80% reduction in HPWL and reduces runtime by 20.24%.

I. INTRODUCTION

With the continuous shrinkage of feature sizes in modern Very Large-Scale Integration (VLSI) and the increasing circuit complexity, timing closure faces growing challenges [1]. To ensure that systems function as intended, designers must prevent timing violations, with metrics such as Worst Negative Slack (WNS) and Total Negative Slack (TNS) serving as indicators of timing performance [2] [3]. Placement, a key stage in VLSI design, determines cell locations and impacts wirelength, power, and timing [4]. Hence, optimizing timing during placement is essential for sign-off [5].

Placement can be divided into global placement (GP), legalization (LG) and detailed placement (DP) stages, and timing optimization can be applied to all these stages. In this flow, GP determines cell positions but may lead to overlaps. LG aligns cells with rows and sites to resolve these overlaps, while DP refines the layout, ensuring legal positions and performing direction-specific optimizations [6] [7].

Many approaches have been proposed for timing-driven detailed placement. Huang et al. [8] mitigated timing violations using Elastic Histogram Compression. Kim et al. [9] minimized clock arrival times in FFs, which significantly alleviated Early violations but also exacerbated Late violations, while other studies [10] [11] concentrated on analytical placement of FFs, gates, and Local Clock Buffer (LCB), improving the quality of the solution through buffer movements. In addition, the research [12]–[15] used Lagrangian relaxation-based methods to guide cell repositioning.

Previous studies have faced performance limitations due to the high runtime overhead introduced by frequent timing analysis calls, the lack of timing compensation for distant regions, and the impact of dual-spacing effects. To address these issues, we introduce a simplified RC-tree and timing–wirelength mapping model that enables timing optimization without cell movements. In addition, we incorporate gradient information into the heuristic framework and enhance it with local reinforcement to ensure both effectiveness and efficiency. Our main contributions can be summarized as follows:

- **Simplified RC-tree for wirelength–timing mapping.** We establish a mapping between wirelength and timing through a simplified RC-tree, while dynamically reconfiguring the LCB–FF topology via routing compensation. This mitigates local search limitations and exploits cross-region placement resources for enhanced optimization.
- **Gradient-guided heuristic algorithm for balanced search.** We integrate gradient-based local optimization with heuristic global exploration, formulating a pin-criticality–driven cost function that balances exploration and exploitation for effective placement optimization.
- **Window-based timing optimization for hotspot cells.** Based on window partitioning and incremental timing updates, we construct a timing-aware cost matrix to achieve optimal cell-to-grid matching, enabling collaborative optimization of timing hotspots.
- Compared to the state-of-the-art detailed placer, our algorithm eliminates Early timing violations in 7 of 8 cases and improves Late slack. Compared to the overall placement framework DREAMPlace 4.0, it achieves average gains of 19.60% in WNS, 55.74% in TNS, while also being 20.24% faster with better HPWL quality.

The remainder of this paper is organised as follows. Section II formulates the optimisation problem. Section III details our proposed methodology. Section IV presents the experimental results. Finally, Section V concludes the paper.

II. PRELIMINARIES

Static Timing Analysis (STA) is essential for verifying whether a circuit meets timing requirements [16]. It models the circuit as a directed acyclic graph (DAG) with alternating cell and net arcs [17], where nodes store timing information and arc directions reflect signal propagation, which introduces

[†]Equal contribution.

*Corresponding author.

delay, slew, and skew. STA primarily includes path-based (PB) and graph-based (GB) methods [18]: PB-STA evaluates each path individually for high accuracy but at high computational cost, whereas GB-STA approximates timing using worst-case endpoint values, offering faster but less accurate analysis; thus, GB-STA is typically used for early evaluations, with PB-STA applied later for refined results.

During the timing-driven placement (TDP) flow, timing performance metrics are typically represented by slack [19], which is calculated using the required arrival time (rat) and arrival time (at) at each endpoint, intuitively expressed as

$$\text{slack}(ep) = \text{rat}(ep) - \text{at}(ep). \quad (1)$$

However, to ensure the design meets timing constraints under different Process, Voltage, and Temperature (PVT) conditions, we often need to load multiple .lib files (typically Early and Late) to calculate the worst-case slack value [16]. Specifically, to meet timing constraints, it is necessary to ensure that signals are ready a certain period of time before being captured (referred to as setup constraints) and that they remain stable for a certain period after being captured (referred to as hold constraints), as

$$\begin{aligned} \text{slack}_{\text{setup}}(ep) &= \text{rat}_{\text{min}}(ep) - \text{at}_{\text{max}}(ep), \\ \text{slack}_{\text{hold}}(ep) &= \text{at}_{\text{min}}(ep) - \text{rat}_{\text{max}}(ep). \end{aligned} \quad (2)$$

Furthermore, in the task of timing optimization, we can further extend the above formula to obtain WNS and TNS to represent global-level timing metrics, as

$$\text{WNS} = \min \{ \text{slack}_{\text{setup/hold}}(n) \mid n \in \text{endpoints} \}, \quad (3)$$

$$\text{TNS} = \sum_{n \in \text{endpoints}} \min(0, \text{slack}_{\text{setup/hold}}(n)). \quad (4)$$

To achieve better performance, it is essential to integrate timing metrics within the detailed placement objective following global placement and legalization [19]. By inputting the legalized placement into a timing-driven detailed placer, cell positions are adjusted within legalization constraints and displacement limits. This approach combines timing metrics with positional information to define the problem:

$$\begin{aligned} \min \quad & \sum_{(x_i, y_i) \in \text{Netlist}} \omega(x_i, y_i) \\ \text{s.t.} \quad & \forall (x_i, y_i), (x_j, y_j) \in \text{Netlist}, \\ & (i < j) \wedge (y_i = y_j) \Rightarrow (x_i + \text{wid}_i < x_j), \end{aligned} \quad (5)$$

where the value of $\omega(x_i, y_i)$ is

$$\max(0, -\text{slack}_{(x_i, y_i)}^{\text{Early}}) + \max(0, -\text{slack}_{(x_i, y_i)}^{\text{Late}}), \quad (6)$$

(x_i, y_i) represents the coordinates of each cell and wid_i represents the width of the i -th cell.

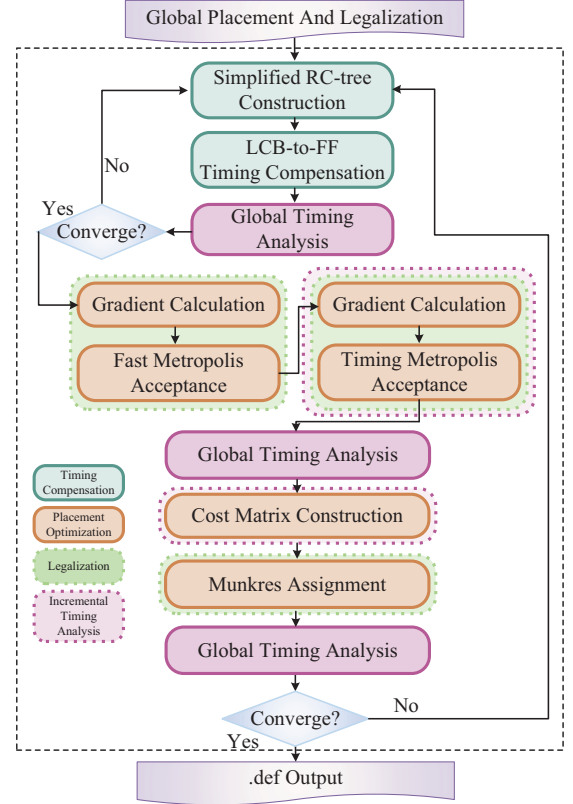


Fig. 1: Overall flow of our algorithm, where the dashed boxes represent auxiliary tools invoked after completing the tasks in the corresponding solid boxes.

III. METHODOLOGY

In this section, we propose a timing-compensated detailed placement framework, as illustrated in Fig. 1. Based on slack calculation, we first apply wire compensation to confine the impact of flip-flops within a reasonable range, followed by iterative placement. The placement process consists of a gradient-guided global heuristic, incremental timing analysis, and local window reinforcement for hotspot cells. These three components complement each other, dynamically balancing exploration and exploitation, thereby guiding timing convergence more effectively.

A. LCB Reallocation Based on Wire Compensation

In STA, slack is computed from flip-flop, which sets the Rat [20]; thus, timing optimization first adjusts the Rat to a proper range. However, conventional scheduling methods only consider the impact of cell load variations on transition rates, which is merely an indirect factor influencing timing. This approach inevitably introduces substantial additional computational overhead, significantly slowing the convergence of placement algorithms. Moreover, traditional methods often neglect the effect of additional interconnects introduced by redistribution, despite such information being critical for accurately deriving the mapping function between wirelength and timing. To address these limitations, we propose a topology

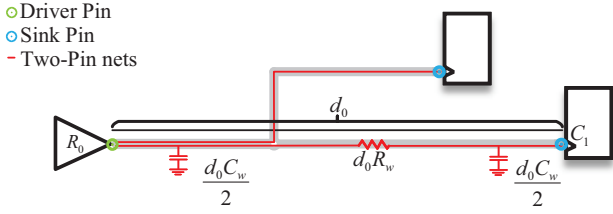


Fig. 2: An example of a simplified RC-tree: a three-pin net is split into two two-pin nets, with R_0 as driver resistance, C_1 as sink load, d_0 as wirelength, and R_w , C_w as unit-length resistance and capacitance.

reconstruction mechanism. By reasonably re-pairing LCB–FF connections, the method effectively compensates for timing.

Firstly, we extract the timing information of all flip-flops, including their d-pin and q-pin, and classify them based on the presence of violations and the type (setup/hold). This classification initially results in 2^4 groups. Building on this, we divide the above 16 groups into three categories based on the compensatory characteristics of timing and wirelength:

(1) **Neutral Type** (Flip-flops without violations): This type of flip-flop possesses slack that can be traded, thereby potentially enlarging the feasible solution space under the constraint of not introducing new timing violations.

(2) **Push-Away Type** (Setup violation on the D-pin or hold violation on the Q-pin): For this type, we select an LCB positioned farther from the initial flip-flop.

(3) **Pull-Close Type** (Hold violation on the D-pin or setup violation on the Q-pin): This type is the opposite of the Push-Away Type, requiring the LCB to be placed closer.

When both D and Q pins violate timing, the smaller value is used as the criterion. Flip-flops with all positive values are sorted descendingly, while those with any negative value are sorted ascendingly to prioritize higher-criticality nodes. Considering computational complexity and accuracy, we construct a simplified RC-tree model (Fig. 2), using the Elmore delay model [21] for 2-pin nets and decomposing multipin nets into 2-pin segments. Within this structure, we establish the mapping between required wirelength adjustments d_0 and timing optimization targets D , as

$$\begin{aligned} D &= R_0 (C_1 + d_0 C_w) + d_0 R_w \left(C_1 + \frac{d_0 C_w}{2} \right) \\ &= R_0 C_1 + R_0 d_0 C_w + d_0 R_w C_1 + \frac{d_0^2 R_w C_w}{2}, \end{aligned} \quad (7)$$

$$\underbrace{\frac{R_w C_w}{2}}_a d_0^2 + \underbrace{(R_0 C_w + R_w C_1)}_b d_0 + \underbrace{(R_0 C_1 - D)}_c = 0. \quad (8)$$

Subsequently, by excluding the negative wirelength d_0 , we obtain the final mapping relationship:

$$d_{max} = \max\left(0, \frac{-b + \sqrt{b^2 - 4ac}}{2a}\right). \quad (9)$$

For FFs of the Neutral Type, since they exhibit no timing violations, the corresponding flip-flops possess transferable

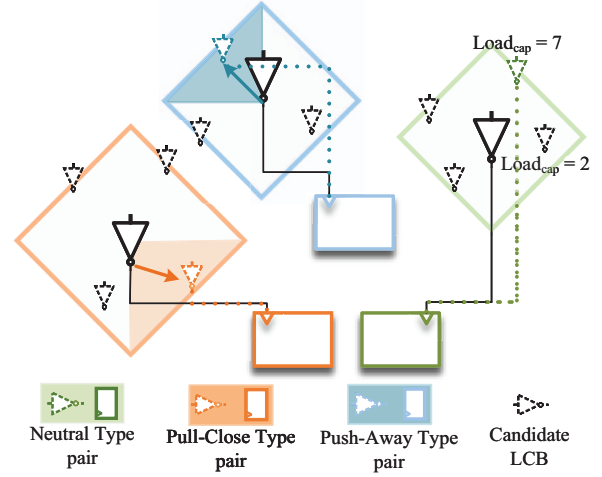


Fig. 3: A simplified example of wire compensation. For efficiency, timing is not updated after every reassignment; instead, the search region is limited to half the computed size, ensuring steady optimization.

timing slack. The amount of slack that can be transferred is defined as the minimum value among the four timing slacks of the cell. This value is then used as the independent variable D in Equation(9) to determine the maximum allowable range d_{max} for topology reconstruction without introducing new timing violations, as illustrated by the green diamond in Fig.3. Within this range, the algorithm selects an LCB as the target buffer if it has the maximum load capacitance, its fan-out is less than $n_{FF}/n_{LCB} + 10$, and none of its fan-out pins exhibit timing violations. The current FF is then reassigned to this buffer. Such a transfer strategy eliminates the need to insert additional LCBs to satisfy potential electrical connectivity constraints of FFs, while simultaneously reducing the number of sibling nodes associated with FFs that actually exhibit timing violations. Moreover, this state aligns well with the requirements of our simplified RC-tree model, minimizing the impact of load capacitances contributed by sibling nodes and thereby enabling more targeted optimization for cells with timing violations.

For the other two types, we collect the minimum positive slack value from the FFs, which is taken as the maximum compensable time of the corresponding net. Using Equation(9), we derive the maximum Manhattan radius. Within this range, we select the LCB whose fan-out is smaller than n_{FF}/n_{LCB} and that has the minimum load capacitance as the target LCB. For push-away type FFs, the selection strategy favors LCBs located closer to the boundary of the Manhattan ring within the quadrants where the current FF is likely to move farther, as illustrated by the blue diamond in Fig.3. By symmetry, the strategy for pull-close type FFs can be derived in an analogous manner, as illustrated by the orange diamond in Fig.3.

B. Timing-Aware Gradient Annealing

In FPGA design, prefabricated logic blocks and interconnects greatly reduce constraint complexity, allowing black-

box algorithms such as simulated annealing to often suffice. In contrast, VLSI design must adhere to strict physical rules and account for clock trees, process corners, and voltage/temperature variations, where each cell movement incurs significant overhead to verify whether a true optimization has been achieved. Moreover, the million-gate scale of modern VLSI circuits makes random search strategies that lead to unsatisfactory performance, as they cannot balance exploration and exploitation. To address these challenges, we propose the Timing-Aware Gradient Annealing (TAGA) algorithm, which incorporates a timing-aware gradient guidance mechanism together with fast local cost evaluation, overcoming the efficiency bottleneck of conventional simulated annealing in timing-driven placement.

The core idea of the algorithm is to combine the local optimization capability of gradient descent with the global search characteristics of simulated annealing. By employing a timing-criticality-driven adaptive search strategy, the algorithm enables efficient exploration of the solution space. Specifically, it first computes the pin importance of each cell as

$$Cri_{pin} = \begin{cases} \max(0, \frac{\min(0, slack)}{WNS}), & WNS < 0 \\ 0, & otherwise \end{cases} \quad (10)$$

$$Cen_{pin} = \begin{cases} Cri_{pin}, & pin \in endpoint \\ \sum Cen_{sink}, & pin \in driver \end{cases} \quad (11)$$

$$pin_{imp} = \frac{Cen_{pin} + Cri_{pin}}{2}. \quad (12)$$

Subsequently, the algorithm adopts a hierarchical optimization scheme to balance computational efficiency and optimization accuracy. In the fast simulated annealing (SA) stage, a lightweight local cost function is employed as

$$C_{fast}(cell, pos) = 0.1 \cdot C_{displacement} + C_{position} \quad (13)$$

where

$$C_{displacement} = \frac{\|pos - pos_{original}\|_1}{D_{max}} \quad (14)$$

represents the displacement penalty term, D_{max} is the maximum displacement constraint, and

$$C_{pos} = \sum_{pin \in cell} \sum_{Pin \in net(pin)} \text{dist}(pos, pos_{Pin}) \cdot 0.001 \quad (15)$$

denotes the position-based cost. In the precise SA stage, a complete timing-aware cost function C_{timing} is employed. For Late optimization, according to Equation(2), the objective is to increase $slack_{setup}$ that the value of at must be reduced when rat remains unchanged. Therefore, our timing cost function is defined as:

$$C_{timing}^{late} = \sum_{pin \in Pins} \sum_{sink \in net(pin, SINK)} pin_{imp}(sink) \cdot \max(at_{fall}(sink), at_{rise}(sink)), \quad (16)$$

where $Pins$ denotes all the pins of the moved cell, and $net(pin, SINK)$ denotes the pin -pins of the net to which

pin belongs. For the Early optimization mode, the situation is the exact opposite of the Late case. According to Equation(2), it is necessary to increase the value of at . Therefore, we define the timing cost as the negative of the computed maximum arrival time at the sink.

Next, we propose a timing-aware gradient-guided neighborhood generation mechanism. Using the fast cost function C_{fast} , which requires no invocation of timing analysis, we compute a numerically stable and computationally inexpensive finite-difference gradient:

$$\nabla_x C = \frac{C_{fast}(pos + \epsilon \hat{x}) - C_{fast}(pos)}{\epsilon} \quad (17)$$

$$\nabla_y C = \frac{C_{fast}(pos + \epsilon \hat{y}) - C_{fast}(pos)}{\epsilon} \quad (18)$$

where $\epsilon = 2$ denotes the finite-difference step size used for numerical approximation of partial derivatives, \hat{x} and \hat{y} is the unit vector in the X and Y direction. After normalization, these gradients are used to generate candidate positions as

$$pos' = clip(pos - 0.6T D_{max} \cdot \frac{\nabla C}{\|\nabla C\|_2} + 0.4T D_{max} \cdot \mathbf{u}), \quad (19)$$

where $clip$ denotes projecting any point outside the placement boundary back to the nearest feasible boundary point, T is the current annealing temperature, and \mathbf{u} is a random unit vector used to preserve the exploratory nature of annealing. If the displacement constraint is violated, the candidate position is scaled back into the feasible region:

$$pos' \leftarrow pos_{orig} + \frac{D_{max}}{\|pos' - pos_{orig}\|_1} (pos' - pos_{orig}). \quad (20)$$

In practice, cells are prioritized by severity of violation, with the most critical ones entering the fast SA phase first, where the acceptance criterion is determined by the differential ΔC of the lightweight cost function C_{fast} . Subsequently, in the precise phase, gradient computation remains unchanged, but the evaluation employs the timing-aware cost function C_{timing} . In this way, the inexpensive and smooth C_{fast} efficiently guides the solution toward promising basins, while the timing-directed cost enforces strict acceptance control. In both phases, we adopt a modified Metropolis acceptance rule:

$$P(accept) = \begin{cases} 1, & \Delta C \leq 0 \\ \exp(-\Delta C / (T \cdot D_{max})), & otherwise \end{cases} \quad (21)$$

with the temperature updated by exponential cooling:

$$T_{k+1} = d_T \cdot T_k \quad (22)$$

where in our implementation $T_0 = 0.8$, $d_T = 0.95$, and the termination threshold is set to $\epsilon = 10^{-6}$. Incorporating displacement scale into normalization suppresses large moves and favors smaller ones, adaptively balancing exploration and exploitation under displacement constraints and improving convergence stability. After each cell iteration, local legalization is performed [22], and in the precise phase an extra local timing update is carried out. The algorithm terminates

when either no improvement is observed over five consecutive iterations or when the WNS reaches zero.

C. Hotspot Cell Window Refinement

After the timing-aware gradient annealing phase, the algorithm outputs a sequence of cells that have been successfully placed in relatively good positions, which can be regarded as a set of locations with both practical movability and potential timing benefits. However, due to the pointwise acceptance and random perturbations inherent in SA, the resulting solutions often lack sufficient coordination. To address this issue, we introduce a refinement stage applied to the sequence of accepted cells $Cell = \{c_1, c_2, \dots\}$ that is partitioned into subsets of at most $K = 5$ cells (an empirically determined hyperparameter) $\mathcal{P} = \{p_1, \dots, p_T\}$ with $|p_i| \leq K \forall i$ and then optimized jointly within their minimum bounding box. The objective of the algorithm within a placement window is to find an optimal one-to-one assignment between a set of cells and the set of available grid sites, such that the total cost is minimized.

To achieve this, our method follows several key steps. First, the geometric positions (x_j, y_j) of the cells within the current subset p_i are extracted and their minimum bounding box aligned with the axis is constructed. The bounding box is then expanded in both x and y directions by one site width and row height, respectively, thereby forming the placement window Win . Within Win , a complete grid \mathcal{G} is generated by scanning with a step size of $(Wid_{site}, Height_{row})$:

$$\mathcal{G} = \{g_1, g_2, \dots, g_M\}. \quad (23)$$

To maintain consistency, we continue to use the cost function $C_{timing}^{late/early}$ in Equation(16). For any cell c_i and candidate site g_i , we define the incremental cost of a tentative move as

$$\Delta J(c, g) = J_{new}(c \rightarrow g) - J_{old}(c), \quad (24)$$

where $J_{old}(c)$ denotes the cost of the cell at its original position, and $J_{new}(c \rightarrow g)$ represents the cost obtained through the incremental timing update when c is temporarily moved to g . Based on this, we construct the cost matrix

$$Cost_{i,j} = \Delta J(c_i, g_j), \quad c_i \in Cell, g_j \in \mathcal{G}. \quad (25)$$

When the number of candidate sites does not match the number of cells to be assigned (typically, the number of sites is much larger), the cost matrix must be expanded to a square matrix $M \times M$, and the expansion follows a specific rule: assign a large penalty BIG to “real rows \rightarrow dummy columns” and 0 to “dummy rows \rightarrow any columns,” ensuring that the optimality of real rows is not affected. The penalty value BIG is defined as

$$BIG = n \cdot (max_{real_cost}) + 1, \quad (26)$$

where n is the number of cells to be assigned, and max_{real_cost} refers to the largest cost among all entries in the cost matrix that map actual cells to actual candidate sites. And Munkres algorithm is then applied to obtain the optimal assignment. Each matched cell is moved to its target white

TABLE I: ICCAD 2015 contest benchmark statistics.

Circuit	Nodes	Flops	LCBs	Nets	Max. Disp
sb1	1209716	144266	7213	1215710	400
sb3	1213253	167923	8396	1224979	400
sb4	795645	176895	8843	802513	400
sb5	1086888	114103	5704	1100825	400
sb7	1931639	270219	13510	1933945	500
sb10	1876103	241267	12063	1898119	500
sb16	981559	142543	7126	999902	400
sb18	768068	103544	5177	771542	400

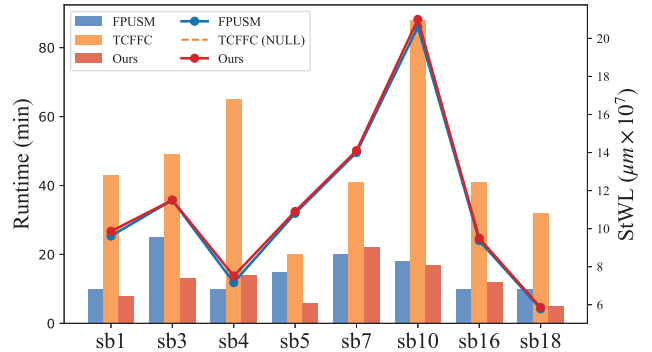


Fig. 4: Comparison of runtime (bar) and STWL (line).

space and legalized via a nearest-empty scheme. After all windows are processed, a global timing analysis updates QoR to capture optimization gains.

IV. EXPERIMENTAL RESULT

We implemented our flow using the open source framework [23], which supports the traversal, legalization, and incremental timing analysis. And tested it on ICCAD 2015 contest benchmarks [24] with the official timer [25]. Table I summarizes the netlist. The experiments were carried out on an Ubuntu machine with a 3GHz CPU and 128GB RAM.

To validate the effectiveness and efficiency of our algorithm, we designed a three-part experiment. First, we compare our algorithm with mainstream detailed placement algorithms on the same task. Next, we integrate our framework with the state-of-the-art open source global placer DREAMPlace 4.0 [7] to assess its optimization capacity under different initial placements and compare it with the global-to-detailed placement co-optimization version of DREAMPlace 4.0 [12]. Finally, we conduct robustness tests on various benchmarks to analyze our algorithm’s generalization ability and optimization stability.

A. Optimization results of detailed placement

Table II presents a comprehensive comparison between our algorithm and state-of-the-art detailed placer, all of which utilize the initial global placement solution provided by ICCAD contest. As shown in Table II, our placer achieves strong performance across most benchmarks. Notably, we eliminated early timing violations in 7 out of 8 cases.

In addition to timing, we evaluated runtime and Steiner wirelength (StWL) as shown in Fig.4, omitting HPWL since prior work only reports StWL. Although identical hardware

TABLE II: Results of the FPUSM [9], TCFFC [15], and ours, with evaluation metrics consistent with previous works.

Benchmark	FPUSM	TCFFC	Ours	FPUSM	TCFFC	Ours	FPUSM	TCFFC	Ours	FPUSM	TCFFC	Ours
	Late WNS(<i>ns</i>)			Late TNS(<i>ns</i>)			Early WNS(<i>ps</i>)			Early TNS(<i>ps</i>)		
sb1	-4.57	-4.42	-4.38	-351.21	-323.94	-329.31	-0.43	0.00	0.00	-0.43	0.00	0.00
sb3	-8.70	-8.27	-8.07	-1160.07	-881.59	-914.10	-5.54	-3.29	0.00	-29.05	-16.50	0.00
sb4	-5.76	-5.60	-5.54	-2462.93	-2309.54	-2341.93	0.00	-3.13	0.00	0.00	-13.80	0.00
sb5	-24.29	-24.70	-24.36	-5842.28	-6327.55	-5943.82	-36.77	-12.24	0.00	-268.60	-54.01	0.00
sb7	-15.21	-15.21	-15.21	-1510.79	-1454.46	-1331.84	-6.38	-7.35	-6.92	-1858.48	-1820.90	-1934.50
sb10	-16.07	-16.13	-15.41	-31518.00	-29445.10	-27725.59	-2.20	-2.40	0.00	-3.47	-12.50	0.00
sb16	-3.84	-3.35	-3.45	-265.57	-209.59	-203.71	0.00	-1.85	0.00	0.00	-1.85	0.00
sb18	-3.81	-3.80	-3.69	-775.87	-701.43	-594.05	0.00	-0.02	0.00	0.00	-0.02	0.00

TABLE III: The comparison is made against the state-of-the-art placer DREAMPlace 4.0 (GP+DP) [12] for WNS (*ps*), TNS (*ps*), max. disp. (μ m), HPWL (μ m), and runtime (*s*).

Benchmark	DREAMPlace 4.0 (GP+DP) [†] [12]					DREAMPlace 4.0 (GP) [7]+OURS (DP)				
	WNS	TNS	max.disp.	HPWL	Runtime*	WNS	TNS	max.disp.	HPWL	Runtime
sb1	-15087	-6789430	6869	8.57E+07	872	-16658	-2545405	6730	9.15E+07	806.5
sb3	-19020	-5698360	7706	9.38E+07	1049	-17031	-2998619	6474	9.18E+07	753.75
sb4	-14098	-14709880	6106	6.62E+07	598	-10330	-7422230	5345	6.34E+07	403.519
sb5	-26483	-10591310	7464	1.05E+08	798	-23606	-4887331	7458	1.01E+08	449.3935
sb7	-17479	-5767843	8819	1.18E+08	1312	-15465	-2431461	8365	1.14E+08	1767.325
sb10	-25554	-64370109	8986	1.95E+08	2669	-19325	-43349911	8194	2.06E+08	1902.34
sb16	-12785	-6402584	7372	9.12E+07	1135	-6867	-1063104	7293	8.77E+07	884.2505
sb18	-12828	-4866836	5098	4.84E+07	563	-8083	-2006766	5068	4.74E+07	372.167
Average Ratio	1.30x	2.63x	1.07x	1.01x	1.33x	1.00x	1.00x	1.00x	1.00x	1.00x

*RT: The runtime of [12] is scaled to reflect the machine difference: the runtime of [7] on our machine $\times \frac{GP+DP \text{ in [12]}}{GP \text{ in [12]}}$.

[†] For DREAMPlace 4.0 [12], we acquire the DEFs from its authors to evaluate.

experiments were not feasible, evaluating physical metrics remains crucial. Our method achieves markedly faster runtime than TCFFC [15], despite running on a CPU with 0.6 GHz lower frequency. Since [15] does not report wirelength, we compare with another state-of-the-art placer, demonstrating clear performance gains with only minor wirelength variations.

B. Results of Optimization with Global Placement

Table III compares the proposed framework with the state-of-the-art timing-driven placer DREAMPlace 4.0 (including GP and DP) [12]. Following [12], we input the eight initial placement solutions into the advanced open-source placer [7], perform global placement, and then apply our method for further timing optimization. Compared with the DEF files from [12], our algorithm achieves substantial improvements in WNS (19.60% on average) and TNS (55.74% on average), while causing minimal disruption to the global placement; the maximum displacement is on average 5.77% smaller than DREAMPlace 4.0 [12]. Table III also presents runtime and HPWL comparisons, showing that our method reduces runtime by 20.24% and lowers HPWL by 0.80% on average, indicating that timing improvements come virtually without cost.

C. Robustness experimental results

To evaluate the robustness of our framework, we replaced the original annealing parameters with a random seed and ran it 10 times across various benchmarks to assess. As shown in Fig.5, even under randomized parameters, our algorithm consistently maintains performance within an acceptable error range. Due to page limitations, we present only the Late TNS data, where the most significant variations were observed, but similar stability trends were also verified in other metrics.

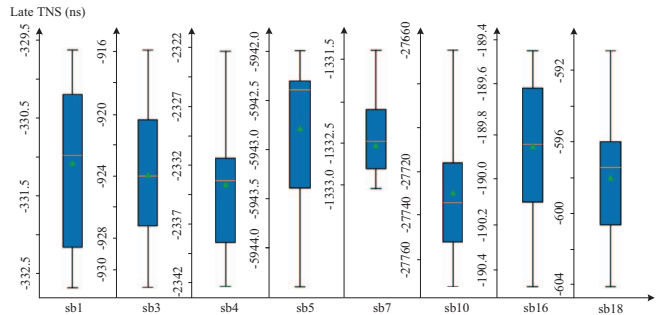


Fig. 5: The Table II data were obtained using the same parameters reported in the papers, which may not suit all cases, causing slight differences—hence our robustness analysis. The orange short line shows the median, and the green triangle indicates the mean.

V. CONCLUSIONS

This paper presents a timing-driven detailed placement framework. It enhances timing metrics through collaborative optimization of LCB reallocation, gradient annealing, and hotspot cell refinement while addressing high cell density and restricted search space issues, offering an effective solution to timing closure challenges in complex VLSI design.

VI. ACKNOWLEDGMENT

This work is supported by the project of the Ministry of Industry and Information Technology High-Quality Development Program (No. CEIEC-2024-ZM02-0056), the 2023 Central Guiding Local Science and Technology Development Fund Project (Grant No. 2023ZYDF074), 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*, 2012, pp. 275–282.
- [2] J. Hu, G. Schaeffer, and V. Garg, "Tau 2015 contest on incremental timing analysis," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 882–889.
- [3] T.-W. Huang, G. Guo, C.-X. Lin, and M. D. Wong, "Opentimer v2: A new parallel incremental timing analysis engine," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 40, no. 4, pp. 776–789, 2020.
- [4] Q. Yanghua, H. Ng, and N. Kapre, "Boosting convergence of timing closure using feature selection in a learning-driven approach," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–9.
- [5] Y. Shi, S. Xu, S. Kai, X. Lin, K. Xue, M. Yuan, and C. Qian, "Timing-driven global placement by efficient critical path extraction," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.
- [6] T. Ajayi and D. Blaauw, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," in *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019.
- [7] P. Liao, S. Liu, Z. Chen, W. Lv, Y. Lin, and B. Yu, "Dreamplace 4.0: Timing-driven global placement with momentum-based net weighting," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 939–944.
- [8] C.-C. Huang, Y.-C. Liu, Y.-S. Lu, Y.-C. Kuo, Y.-W. Chang, and S.-Y. Kuo, "Timing-driven cell placement optimization for early slack histogram compression," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.
- [9] S. Kim, S. Do, and S. Kang, "Fast predictive useful skew methodology for timing-driven placement optimization," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.
- [10] G. Flach, J. Monteiro, M. Fogaça, J. Puget, P. Butzen, M. Johann, and R. Reis, "An incremental timing-driven flow using quadratic formulation for detailed placement," in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2015, pp. 1–6.
- [11] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis, "Drive strength aware cell movement techniques for timing driven placement," in *Proceedings of the 2016 International Symposium on Physical Design*, 2016, pp. 73–80.
- [12] 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.
- [13] C. Guth, V. Livramento, R. Netto, R. Fonseca, J. L. Güntzel, and L. Santos, "Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 141–148.
- [14] G. Wu and C. Chu, "Two approaches for timing-driven placement by lagrangian relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 12, pp. 2093–2105, 2017.
- [15] D. Mangiras, A. Stefanidis, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2835–2848, 2019.
- [16] J. Bhasker and R. Chadha, *Static timing analysis for nanometer designs: A practical approach*. Springer Science & Business Media, 2009.
- [17] O. CIRCUITS, "Timing analysis and optimization of sequential circuits," 1999.
- [18] Z. Guo and Y. Lin, "Differentiable-timing-driven global placement," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1315–1320.
- [19] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," *Handbook of Algorithms for Physical Design Automation*, pp. 423–446, 2008.
- [20] J. Hu, G. Schaeffer, and V. Garg, "Tau 2015 contest on incremental timing analysis," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 882–889.
- [21] T.-W. Huang, G. Guo, C.-X. Lin, and M. D. Wong, "Opentimer v2: A new parallel incremental timing analysis engine," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 40, no. 4, pp. 776–789, 2020.
- [22] J. C. Puget, G. Flach, M. Johann, and R. Reis, "Jezz: An effective legalization algorithm for minimum displacement," in *Proceedings of the 28th Symposium on Integrated Circuits and Systems Design*, 2015, pp. 1–5.
- [23] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis, "Rsyn: An extensible physical synthesis framework," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 33–40.
- [24] N. Viswanathan, S.-H. Huang, R.-B. Lin, and M.-C. Kim, "Overview of the 2015 cad contest at iccad," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 910–911.
- [25] 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)*. IEEE, 2015, pp. 921–926.