

Timing-driven Detailed Placement via TimingMask-guided Path-level Optimization

Ruo-Tong Chen^{1,2,3}, Chengrui Gao^{1,2,3}, Siyuan Xu³, Ke Xue^{1,2}, Yunqi Shi^{1,2}, Xi Lin^{1,2},
Mingxuan Yuan³, Chao Qian^{†1,2}, Zhi-Hua Zhou^{1,2}

¹National Key Laboratory for Novel Software Technology, Nanjing University, China

²School of Artificial Intelligence, Nanjing University, China

³Huawei Noah's Ark Lab, China

Abstract—Timing-driven detailed placement is a critical stage in very large scale integrated (VLSI) design, aiming to locally adjust cell positions to further improve circuit timing performance. Existing methods commonly adopt proxy metrics as optimization objectives, such as weighted wirelength and approximate delay. However, these surrogate metrics are not fully aligned with the final timing metrics obtained through static timing analysis (STA), often leading to suboptimal timing results. Besides, methods based directly on STA tools suffer from very low search efficiency, making the cost of timing optimization prohibitive. To address these issues, we propose an effective timing-driven detailed placement method via TimingMask-guided path-level optimization. One core of our method is the *TimingMask* guidance mechanism, which integrates both arc delay and path slack information based on the RC timing model, thereby providing more targeted and effective guidance for refinement of critical cells. Meanwhile, our method adopts a path-level timing evaluation strategy with incremental updates, accelerating the optimization process while preserving timing accuracy. Experimental results on the ICCAD 2015 contest benchmarks demonstrate that our method significantly outperforms state-of-the-art detailed placement methods such as DREAMPlace4.0 DP, achieving an average improvement of 25.3% in total negative slack (TNS) and 21.7% in worst negative slack (WNS).

Index Terms—Physical Design, Detailed Placement, Timing-Driven Placement.

I. INTRODUCTION

The exponential growth in transistor density and continuous downscaling of standard cells within very large scale integrated (VLSI) circuits have enhanced the complexity of physical design in modern semiconductor development [1], [2]. As an important stage, placement aims at finding the optimal positions for circuit components by minimizing wirelength and timing violations [3]–[8]. Traditionally, placement is divided into two stages: global placement and detailed placement [9]–[12]. Global placement can produce an overall good placement result, but some approximations made to simplify the problem due to the large problem size may result in some issues, such as inadequate wirelength optimization and timing violations in local regions. Therefore, detailed placement is required to further improve the placement quality locally [13]. Most previous detailed placement methods focus on reducing the total wirelength [13], [14] while ignoring the timing violations, which may bring more complexity to the downstream design stages [15], [16]. In contrast, timing-driven detailed placement

pays attention to the optimization of timing metrics, such as total negative slack (TNS) and worst negative slack (WNS), by perturbing the current placement solution locally around critical paths [17]–[19].

Recently, researchers have been making contributions to solving the timing-driven detailed placement problem, and the methods can be roughly divided into two categories: gradient-based and search-based methods. Gradient-based methods typically translate timing analysis feedbacks into connection weights and use the differentiable weighted-wirelength as their optimization objective [20]–[22]. However, these methods primarily focus on optimizing wirelength, which does not fully align with the timing metrics derived from static timing analysis (STA). Compared with gradient-based methods, search-based methods can define timing-related costs more precisely and enable finer-grained positional adjustments for individual cells, thus becoming the primary research direction [23]–[26]. They first select some candidate positions for each cell, and then choose the best position based on a predefined timing-related cost function. Nevertheless, most search-based methods rely on approximate calculations for their cost functions, limiting the timing accuracy. This issue can be alleviated by performing STA after each cell movement and accepting the move with timing improvement [27], which, however, will incur substantial computational costs due to frequent cell-level STA evaluations.

In order to address the above issue, we propose an effective timing-driven Detailed Placement method via TimingMask-guided Path-level optimization, briefly called TimingPath-DP. In particular, multiple critical paths are simultaneously adjusted with the guidance of a timing-related data structure named TimingMask. TimingMask employs a weighted-squared wirelength to approximate the timing impact, and records this value for each candidate movement of the critical cells along the paths. Then, multiple candidate improved paths are sampled according to TimingMask for each critical path, and the one with the best timing metrics derived from an accelerated STA is selected. Thus, our method directly targets the optimization of timing metrics derived from STA, instead of proxy costs. The path-level timing evaluation alleviates the high computational costs of calling STA, and TimingMask provides targeted and effective search guidance. Experimental results on the ICCAD 2015 contest C benchmark [28] show that our proposed method significantly outperforms state-of-the-art

† Corresponding author. E-mail: qianc@lamda.nju.edu.cn

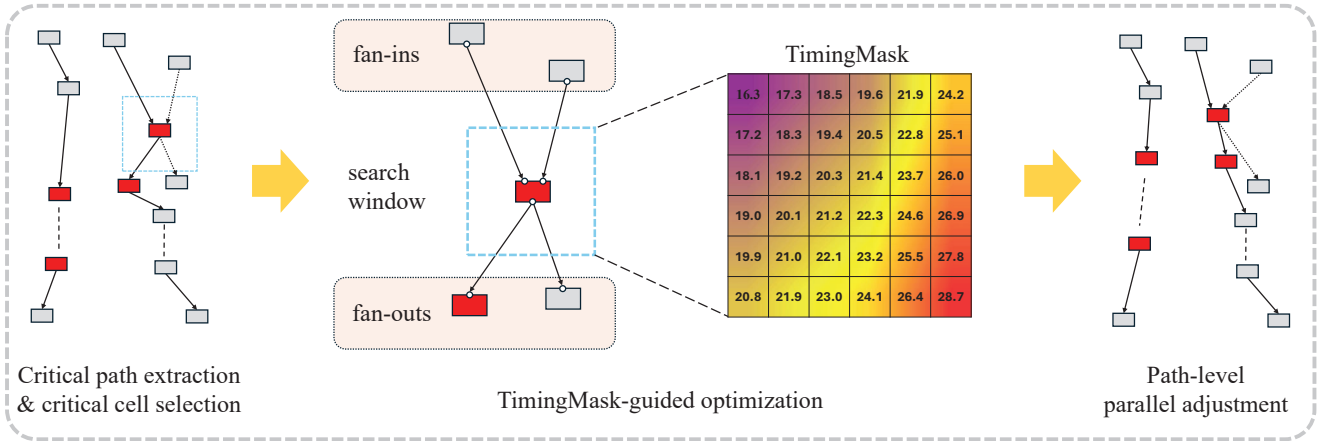


Fig. 1. Flow of our proposed method TimingPath-DP, which is an iterative process. In each iteration, it begins with critical path extraction to identify critical cells (i.e., cells in red) for local refinement within a search window. For each critical cell, we compute a TimingMask, which quickly estimates the timing impact of each candidate movement in the search window. After that, adjustments are conducted in parallel at the path level. That is, multiple candidate improved paths are sampled for each critical path in parallel according to TimingMask, and the one with the best timing improvement (calculated by STA) is selected.

methods (e.g., DREAMPlace 4.0 DP [4]), achieving an average improvement of 25.34% in TNS and 21.72% in WNS. The empirical comparison between TimingMask-guided sampling and uniform sampling for generating candidate improved paths also confirms the effectiveness of using TimingMask. We finally visualize the critical paths before and after optimization, showing that our method significantly improves the uniformity of cell distribution, closely aligned with the RC delay model.

II. BACKGROUND

A. Timing-driven Detailed Placement

The input of a timing-driven detailed placement problem is a netlist $\mathcal{N} = (V, E)$, where V denotes the information (e.g., original position and driving strength) about circuit components (including macros and standard cells), and E is a hyper-graph comprised of hyper-edges, which signifies the interconnectivity of all components [29]. The timing-driven detailed placement problem entails the local adjustment of standard cells to mitigate timing violations, which is usually solved by an iterative process [30]–[32]. Specifically, at the beginning of each iteration, the timer is updated based on the current layout, and several cells on or around critical paths are selected for adjustment [4], [20]. Gradient- or search-based methods can be applied to decide the cell moves at each iteration. A legalized placement layout should be provided as the final output.

B. Static Timing Analysis

Static timing analysis (STA) [33] provides timing simulations for the circuit layout by modeling the design as a directed acyclic graph, where nodes represent circuit pins and edges correspond to timing arcs that define signal propagation directions. STA computes the arrival time t_{at} and the required arrival time t_{rat} through two signal propagation stages: forward and backward propagation [34]. The difference between them at any pin p defines the slack:

$$\text{Slack}(p) = t_{rat}(p) - t_{at}(p).$$

Slack serves as a critical measure of timing quality, with negative values indicating timing violations. We calculate the slack value for each end-point pin and quantify the timing violation using WNS and TNS [35]:

$$\begin{aligned} \text{WNS} &= \min_{p \in V} \min(\text{Slack}(p), 0), \\ \text{TNS} &= \sum_{p \in V} \min(\text{Slack}(p), 0), \end{aligned}$$

which indicates the smallest negative slack and the sum of all negative slacks, respectively. Note that here V represents the set of all end-point pins. If WNS and TNS are equal to zero, all timing constraints are met.

C. Search-based Methods

Existing search-based techniques typically identify a subset of candidate positions for each critical cell, based either on spatial constraints such as a local window [4], or on delay-related heuristics, including Euclidean distance and half-perimeter wirelength (HPWL) [23], [24], [27]. A surrogate objective function is then evaluated for each candidate position to determine the optimal placement. To mitigate computational overhead, this surrogate is often derived from simplified metrics, such as pin-to-pin distances [23], [24], or approximated timing parameters [4], [25]. Georgakidi et al. [23] moved the cell to the position with minimum HPWL. Livramento et al. [24] determined candidate positions based on the Euclidean distance between the cell and its fan-ins or fan-outs, and selected the optimal position with minimum weighted delay. Lee and Li [25] estimated the timing parameters using a regression model and represented the cost function with approximate delays. More recently, Liao et al. [4] efficiently approximated the delay using Elmore delay parameters scaling and 2-hop forward timing propagation, and searched the best position within a local window. However, due to the inherent mismatch between the surrogate cost and the actual timing metrics obtained via STA, these methods often converge to suboptimal solutions.

III. METHOD

The overview of our proposed method TimingPath-DP is illustrated in Fig. 1. It is an iterative optimization process, similar to the evolutionary optimization process [36]. Each iteration contains two important components: TimingMask-guided optimization and Path-level parallel adjustment, which are introduced in Sections III-A and III-B, respectively. After several iterations of adjustment, the circuit layout is legalized by the legalization operator in DREAMPlace [3], and the final placement result is produced.

A. TimingMask-guided Optimization

We first introduce how to approximate the timing impact in an efficient way, and then present how to improve the position of a critical cell using TimingMask guidance.

1) *Fast Estimation of Timing Impact*: To align well with the final timing metrics, we start from the widely adopted RC delay model [37]–[39] to derive our timing impact model. Given a distributed RC network of a net, the delay from the source s to sink t can be calculated as $\text{Delay}_{s \rightarrow t} = R_{s \rightarrow t} C_t$, where $R_{s \rightarrow t}$ represents the equivalent resistance from s to t , and C_t represents the capacitance at node t . Since R and C are both linear to wirelength, the delay grows quadratically. Thus, we choose the squared pin-to-pin Manhattan distance D to approximate the delay between pin p and pin p' :

$$D(p, p') = (|x_p - x_{p'}| + |y_p - y_{p'}|)^2,$$

where (x_p, y_p) and $(x_{p'}, y_{p'})$ denote the position of pin p and pin p' , respectively. Then, the delay impact $Q_c(x, y)$ of a critical cell c on the candidate position (x, y) can be calculated as the summation of the squared wirelength from its fan-ins \mathcal{F}_{in} and fan-outs \mathcal{F}_{out} :

$$Q_c(x, y) = \sum_{p \in \text{Pin}_c} \sum_{p' \in \mathcal{F}_{in} \cup \mathcal{F}_{out}} \mathbb{I}(p, p') \cdot D_{x,y}(p, p'),$$

where Pin_c denotes the pin set of cell c , $\mathbb{I}(p, p')$ is an indicator function that denotes whether a connection exists between pin p and pin p' , and $D_{x,y}(p, p')$ is the squared Manhattan distance between p and p' when the critical cell c is located in (x, y) .

While minimizing delay is a common optimization objective [4], [8], a smaller cumulative delay across a cell's neighbor connections does not inherently guarantee improved timing performance. This is because the required arrival time varies across different primary outputs, leading to differing maximum delay tolerances for each path. As illustrated in Fig. 2, the red connection with a higher delay does not result in a negative slack. Thus, to more accurately evaluate timing criticality, it is essential to take slack into consideration. We first define the slack value of connection (p, p') as the worst slack among all paths passing through it. Then, a pin-to-pin weighting strategy considering both delay and slack is proposed as follows:

$$\omega(p, p') = \text{Normalization}(\text{Delay}_n(p, p') - \text{Slack}_n(p, p')),$$

where $\text{Delay}_n(p, p')$ and $\text{Slack}_n(p, p')$ denote the normalized delay and slack of connection (p, p') , respectively. Higher weights are assigned to critical connections (those with high

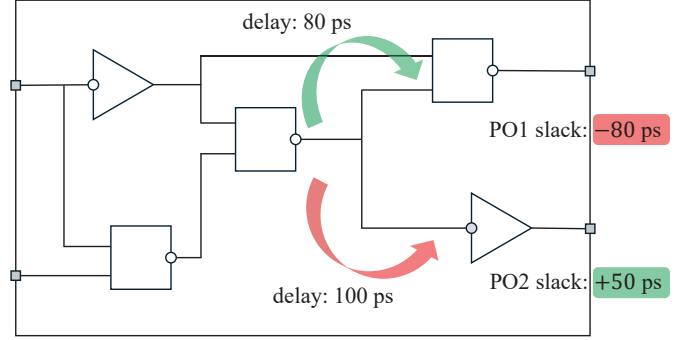


Fig. 2. Illustration of delay and slack on critical paths. Although the green connection has a smaller delay than the red connection, the slack at primary output PO1 is worse than that at PO2.

delay and low slack) to ensure that they are prioritized during optimization. Assisted with this weighting strategy, the timing impact Q_c^ω of a critical cell c on the candidate position (x, y) can be approximated as follows:

$$Q_c^\omega(x, y) = \sum_{p \in \text{Pin}_c} \sum_{p' \in \mathcal{F}_{in} \cup \mathcal{F}_{out}} \mathbb{I}(p, p') \cdot \omega(p, p') \cdot D_{x,y}(p, p').$$

Notably, the weight $\omega(p, p')$ is computed based on the STA report at the beginning of each iteration, which does not vary with different candidate positions. Therefore, Q_c^ω can be computed efficiently for all candidate positions to provide a fast timing impact estimation.

2) *TimingMask Guidance*: Recent works [5], [6] employ a WireMask to guide the optimization of HPWL and demonstrate superior performance. Inspired by them, we design a TimingMask M_c for each critical cell c to record the weighted-squared wirelength among all candidate positions. The selection of cell's candidate positions is implemented by setting a local search window, as shown by the blue box in Fig. 3(a). TimingMask is a matrix with size equal to the search window, and each entry $M_c(i, j)$ represents the estimated timing effect when the cell c is placed at the candidate position $(x_{i,j}, y_{i,j})$:

$$M_c(i, j) = Q_c^\omega(x_{i,j}, y_{i,j}),$$

where (i, j) are matrix indices, and $(x_{i,j}, y_{i,j})$ is the real physical coordinates when the cell c is placed at coordinate (i, j) of the search window. Fig. 3(b) gives an example of TimingMask, where the deeper purple regions (e.g., top-left) correspond to the positions with lower weighted-squared wirelength, suggesting potential timing improvement.

To examine the correlation between TimingMask and the actual timing metrics, we analyze the relationship between weighted-squared wirelength and TNS variation across various locations within the search window. As shown in Fig. 4(a), a negative correlation is generally observed: the positions with smaller weighted-squared wirelength typically yield better timing performance. Thus, to efficiently search for a better position, we employ a sampling strategy leveraging TimingMask, where the sampling probability for each candidate position (i, j) is defined as:

$$P_c(i, j) = \frac{1/M_c(i, j)}{\sum_{i', j'} 1/M_c(i', j')}. \quad (1)$$

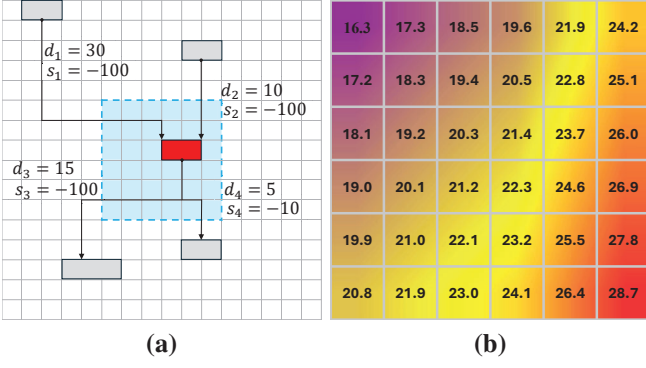


Fig. 3. Illustration of search window and TimingMask. (a) The blue box is the search window of the cell in red. d_i and s_i denote the delay and slack of the corresponding pin-to-pin connection. (b) Visualization of TimingMask for the search window shown in (a). A more purple color indicates a smaller weighted-squared wirelength, while a redder color indicates a larger one.

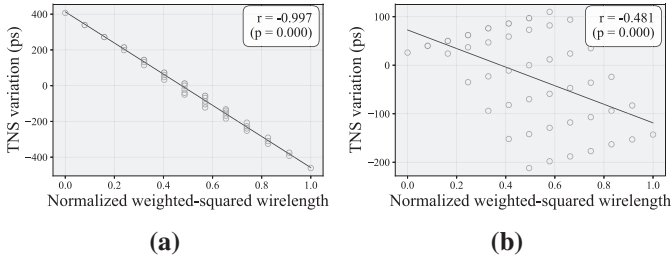


Fig. 4. Illustration of the correlation between normalized weighted-squared wirelength and TNS variation, where r denotes the Pearson correlation coefficient, and p is the significance level. (a) A significant negative linear correlation exists in most cases. (b) The negative correlation can be weak in some cases.

That is, we assign higher probabilities to the positions with smaller weighted-squared wirelength, thereby prioritizing those likely to improve timing. Note that we do not use greedy selection, i.e., directly select the candidate position with the minimal weighted-squared wirelength. This is because in some cases as illustrated in Fig. 4(b), the negative correlation between weighted-squared wirelength and timing can be weak, due to some approximation errors, e.g., the approximation error between Manhattan distance and real wirelength. The sampling strategy is more robust, and can maintain the ability to explore diverse candidate positions and discover improved solutions.

B. Path-level Parallel Adjustment

1) *Critical Path Extraction*: We now introduce the overall optimization flow, which begins with the extraction of critical paths. Previous methods typically extract the top- n worst critical paths [4], [23]. However, the extracted paths tend to concentrate on a limited number of critical endpoints, which not only misaligns with the TNS metric [8], but also brings strong inter-path coupling. To reduce the mutual influence between critical paths, we adopt the extraction method from [8] and incorporate a post-processing procedure for further improvement, as detailed in Algorithm 1. We first extract one critical path for each of the n most critical endpoints using `extract_timing_endpoint($n, 1$)` in line 1. In lines 6–7, cells that appear in previous critical paths are excluded from the current path to avoid repeated adjustment. Since cells with larger delay from their fan-ins have greater potential for optimization, we

Algorithm 1 Critical path extraction

Input: Number n of critical paths, number k of critical cells of each path.

Output: Critical paths \mathcal{P}_{new} .

```

1:  $\mathcal{P}_{origin} \leftarrow \text{extract\_timing\_endpoint}(n, 1)$ ;  $\triangleright$  Extract one
   critical path for each of the  $n$  most critical endpoints [8]
2:  $\mathcal{P}_{new} \leftarrow \emptyset$ ;
3: for each path  $p_{origin} \in \mathcal{P}_{origin}$  do
4:    $p \leftarrow \emptyset$ ;
5:   for each standard cell  $c \in p_{origin}$  do
6:     if  $c \notin \mathcal{P}_{new}$  then
7:        $p.add(c)$ 
8:    $d \leftarrow \text{get\_delay}(p)$ ;  $\triangleright$  Compute delay of each cell
9:    $p_{new} \leftarrow \text{select\_cell}(p, d, k)$ ;
    $\triangleright$  Select  $k$  cells with the worst delays
10:   $\mathcal{P}_{new}.add(p_{new})$ 
11: return  $\mathcal{P}_{new}$ 

```

Algorithm 2 Path-level parallel adjustment

Input: Circuit component positions pos , number n of critical paths, number k of critical cells of each path, number h of sample times.

Output: Updated circuit component positions pos .

```

1:  $TNS_{origin} \leftarrow \text{timer.update}(pos)$ ;  $\triangleright$  Update timing
2:  $\mathcal{P} \leftarrow \text{extract\_critical\_path}(n, k)$  by Algorithm 1;
3:  $\mathcal{S}_{\mathcal{P}} \leftarrow \emptyset$ ;  $\triangleright$  Store candidate solutions for all critical paths
4: for each path  $p \in \mathcal{P}$  in parallel do
5:    $\mathcal{S}_p \leftarrow \text{sample}(p, h)$  according to Eq. (1);
    $\triangleright$  Sample  $h$  solutions in parallel
6:    $\mathcal{S}_{\mathcal{P}}.add(\mathcal{S}_p)$ 
7: for each solution set  $\mathcal{S}_p \in \mathcal{S}_{\mathcal{P}}$  do  $\triangleright$  Path-level evaluation
8:    $TNS_{best} \leftarrow TNS_{origin}$ ;
9:   for each solution  $s \in \mathcal{S}_p$  do
10:     $TNS_s \leftarrow \text{timer.update}(s)$ ;
11:    if  $TNS_s > TNS_{best}$  then
12:       $pos_{best} \leftarrow s$ ;
13:       $TNS_{best} \leftarrow TNS_s$ 
14:     $pos.update(pos_{best})$ 
15: return  $pos$ 

```

select the top k cells with the worst delays on each critical path for adjustment in lines 8–9, which significantly reduces the search space, thereby improving the search efficiency.

2) *Parallel Adjustment*: To facilitate efficient timing refinement, we provide our path-level parallel adjustment method in Algorithm 2, which treats each path as a unit for parallel adjustment and evaluation. Timing is first updated in line 1 at the beginning of each iteration. Then, we extract the critical paths in line 2 and adjust them in parallel to improve the search efficiency. Specifically, each critical path is adjusted following the signal propagation orientation, and a new position is sampled for each cell according to the distribution characterized by Eq. (1). In order to search for a solution with better timing improvement, we perform multiple samplings for each path in lines 4–6. Due to the low intersection among the extracted critical paths, the overall timing impact of them can

be approximated by independent single-path evaluations. Thus, in lines 7–13, we separately evaluate the timing performance of candidate solutions of each critical path, and accept the one with the greatest TNS improvement. To further improve the efficiency, we speed up the path-level STA process by performing an incremental rather than a full timing update during evaluation. It is achieved by only regenerating the Steiner tree for nets associated with the moving cells, which significantly reduces the runtime of both Steiner tree generation and timer update, e.g., by 99.5% and 96.7%, respectively, on the design *superblue1*. In our implementation, we leverage the open-source timer *OpenTimer* [40] to conduct STA, and enhance it for efficient path-level evaluation.

IV. EXPERIMENTS

We use the ICCAD 2015 contest C benchmark suites [28] as our test-bed. All results are evaluated using the official evaluation kit from the ICCAD 2015 contest for fair comparison.

A. Baseline

We compare our proposed method *TimingPath-DP* with four baseline methods: *DREAMPlace4.0 GP* [4], *DREAMPlace4.0 DP* [4], *LocalSearch*, and *TimingPath-DP (uniform)*.

- 1) *DREAMPlace4.0 GP* [4]: A state-of-the-art timing-driven global placer. We employ it to perform global placement for all designs, and utilize the resulting placement as the input for other detailed placement algorithms.
- 2) *DREAMPlace4.0 DP* [4]: A state-of-the-art search-based detailed placer. It derives the cost function using Lagrangian relaxation, and estimates the delay within the search window using Elmore delay parameters scaling. Since this method is not open-sourced, we report the experimental results from the original paper.
- 3) *LocalSearch*: *LocalSearch* is modified from *DREAMPlace4.0 DP* by directly optimizing the timing metrics derived from STA. For each cell to be adjusted, it performs an exhaustive search within the search window to identify the position with the maximum TNS metric.
- 4) *TimingPath-DP (uniform)*: *TimingPath-DP (uniform)* is an ablated version of our method that retains the core search framework, but instead samples the position of each critical cell from a uniform distribution.

The number of timing evaluations is set to 20000 for *LocalSearch*, *TimingPath-DP (uniform)* and *TimingPath-DP*.

B. Main Results

Table I presents a comprehensive comparison of the TNS and WNS metrics between our proposed *TimingPath-DP* and other baseline methods. The results show that our method significantly outperforms all other timing-driven placers, and achieves the best TNS results across all test cases and the best WNS results in seven out of the eight cases. Below, we provide a specific analysis through several comparisons.

Compared to the input layout generated by *DREAMPlace4.0 GP*, *TimingPath-DP* achieves an average improvement of 30.53% (23.75%) on TNS (WNS), confirming the efficacy of our method in incremental timing optimization based on

the GP layout. When compared to *DREAMPlace4.0 DP* (see column 3), *TimingPath-DP* also exhibits significant average improvements of 25.34% and 21.72% on TNS and WNS, respectively. We attribute these enhancements to our strategy of directly optimizing the final timing metrics. In contrast, although *LocalSearch* also adopts the final metrics as its objective function, it yields inferior timing performance compared to *TimingPath-DP (uniform)* due to its cell-level exhaustive search, which suffers from low efficiency compared to our path-level sampling method.

Finally, the comparison between *TimingPath-DP* and its ablated version, *TimingPath-DP (uniform)*, is provided in the last two columns. The results demonstrate an average improvement of 15.47% (21.79%) on TNS (WNS) conferred by the *TimingMask* guidance. This clearly indicates that *TimingMask* can effectively capture timing variation trends and provide more targeted and effective guidance for cell refinement, thereby substantially enhancing search efficiency and leading to better overall performance.

C. Illustration of TNS Convergence Curves

To further verify the search efficiency of *TimingPath-DP*, we compare the TNS convergence curves of all methods in Fig. 5. The number of timing evaluations of different methods is aligned for a fair comparison.

Compared to *DREAMPlace4.0 DP* (denoted by the green dashed line), our method achieves comparable performance after a limited number of timing evaluations. Due to the exponential growth of multipliers, *DREAMPlace4.0 DP* is incapable of further improvement [4], while *TimingPath-DP* can deliver significant timing improvements if more runtime is provided.

Regarding the convergence speed, both *TimingPath-DP (uniform)* and *TimingPath-DP* outperform *LocalSearch*. *LocalSearch* improves slowly since the evaluation for all candidate positions requires a large number of STAs. This result highlights the efficiency of our sampling strategy and path-level evaluation scheme. Furthermore, while *TimingPath-DP (uniform)* yields decent improvement, *TimingPath-DP* converges at a significantly faster rate in all test cases, which demonstrates the critical role of *TimingMask* in efficiently guiding the search toward TNS improvement.

D. Visualization of Critical Paths

We finally visualize the critical paths before and after optimization. Fig. 6 shows an example of the critical paths with WNS in *superblue1*. The cells along the critical path before optimization exhibit a clustered distribution. While the connection distance within each cluster is short, the interconnects between these clusters are excessively long, which introduces significant delays and becomes the primary cause of timing violations. After being optimized by *TimingPath-DP*, the uniformity of distances between connected nodes improves (as highlighted within the yellow ellipses) while maintaining the overall cell distribution. Since delay is quadratic in wirelength, this adjustment substantially shortens the critical long-distance connections, thereby dramatically reducing their delays (e.g., delay is significantly reduced from 3586.46 ps to 1976.80 ps

TABLE I

RESULTS OF TNS ($\times 10^5$ ps) AND WNS ($\times 10^3$ ps) ON EIGHT DESIGNS, WHERE THE PROPOSED METHOD TIMINGPATH-DP IS COMPARED WITH FOUR BASELINE ALGORITHMS, DREAMPLACE 4.0 GP [4], DREAMPLACE 4.0 DP* [4], LOCALSEARCH AND TIMINGPATH-DP (UNIFORM).

| Benchmark | DREAMPlace4.0 GP [4] | | DREAMPlace4.0 DP [4] | | LocalSearch | | TimingPath-DP (uniform) | | TimingPath-DP (ours) | |
|---------------|----------------------|---------------|----------------------|---------------|-------------|---------------|-------------------------|---------------|----------------------|---------------|
| | TNS | WNS | TNS | WNS | TNS | WNS | TNS | WNS | TNS | WNS |
| superblue1 | -83.15 | -13.86 | -67.04 | -13.06 | -73.05 | -12.64 | <u>-60.61</u> | <u>-12.60</u> | -43.36 | -8.52 |
| superblue3 | -58.05 | -16.54 | -52.28 | -16.26 | -57.03 | -15.85 | <u>-53.94</u> | <u>-15.54</u> | -45.95 | -11.98 |
| superblue4 | -143.08 | -12.74 | -136.17 | <u>-11.06</u> | -135.35 | -12.20 | <u>-121.29</u> | <u>-12.20</u> | -86.37 | -7.50 |
| superblue5 | -95.77 | -27.27 | -94.00 | -23.70 | -92.30 | -26.58 | <u>-86.85</u> | <u>-27.22</u> | -79.41 | <u>-26.20</u> |
| superblue7 | -64.93 | -15.22 | -58.37 | -15.22 | -63.83 | -15.22 | <u>-57.42</u> | <u>-15.98</u> | -55.30 | -15.22 |
| superblue10 | -721.93 | -26.66 | -705.80 | -25.38 | -712.66 | <u>-24.92</u> | <u>-668.78</u> | <u>-30.72</u> | -613.29 | -21.33 |
| superblue16 | -122.46 | -8.76 | -117.70 | -11.28 | -120.13 | -8.43 | <u>-76.80</u> | <u>-9.22</u> | -62.70 | -7.22 |
| superblue18 | -44.97 | -12.15 | -43.49 | -11.69 | -42.73 | -10.83 | <u>-30.85</u> | <u>-8.88</u> | -26.91 | -7.14 |
| Average Ratio | 1.50 | 1.36 | 1.39 | 1.33 | 1.33 | 1.33 | 1.20 | 1.31 | 1.00 | 1.00 |

* For DREAMPlace 4.0 DP [4], we borrow the results reported in the original paper as it is not open-source.

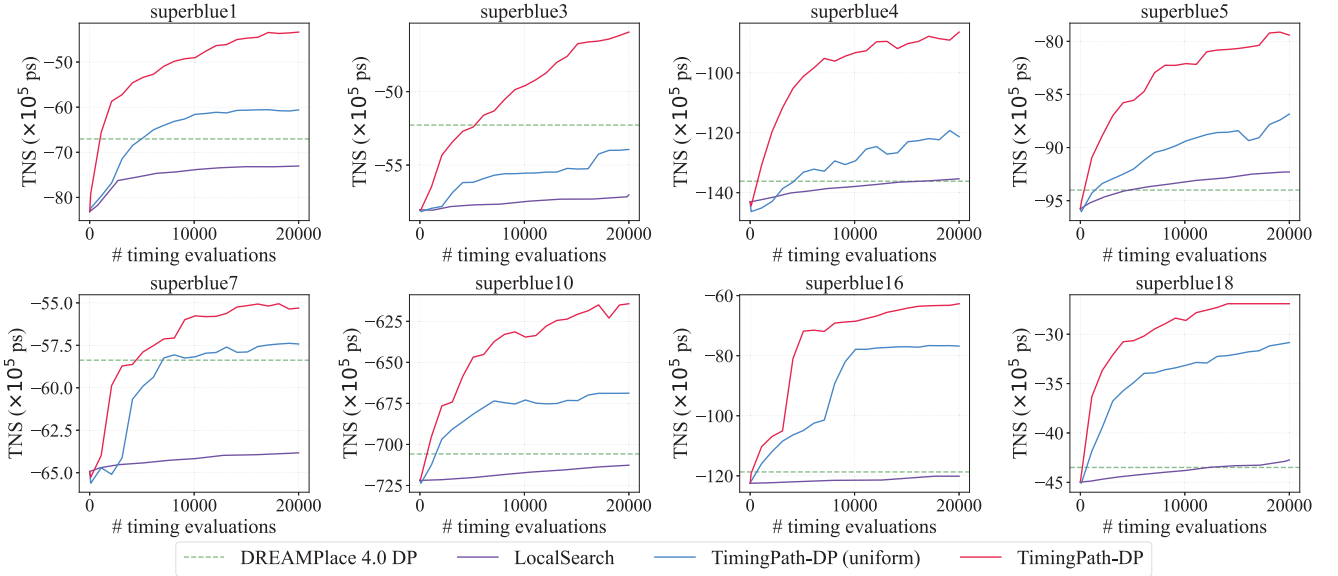


Fig. 5. Illustration of TNS convergence curves.

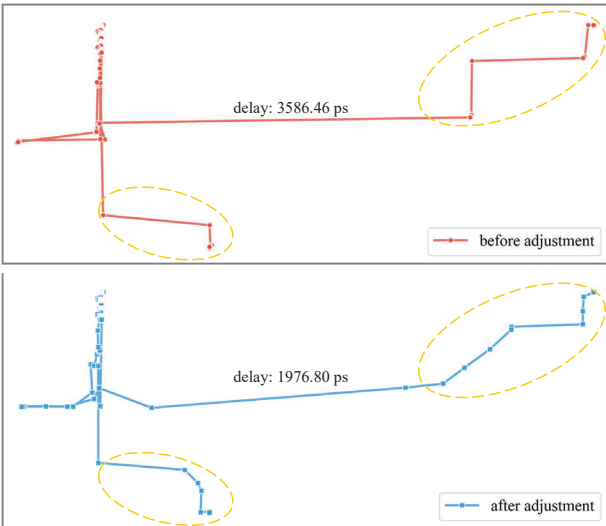


Fig. 6. Illustration of the timing critical paths with WNS of superblue1 before and after adjustment.

for the longest connection shown in Fig. 6) and improving the overall timing performance.

V. CONCLUSION

In this paper, we present an efficient search method that directly targets the optimization of timing metrics derived from STA for timing-driven detailed placement. By using the TimingMask guidance mechanism and path-level timing evaluation strategy, our method significantly improves the search and computational efficiency. Experimental results on the ICCAD 2015 benchmark have shown that our method makes a great improvement in timing quality and search efficiency. In the future, we plan to extend our method to 3D integrated circuits [41], which may need to improve the estimation of the timing impact of 3D critical nets by considering hybrid bonding.

VI. ACKNOWLEDGMENT

This work was supported by the National Science and Technology Major Project (2022ZD0116600), the Fundamental Research Funds for the Central Universities (14380020), and the National Science Foundation of China (62276124, 624B2069).

REFERENCES

- [1] Y. Zhou, Y. Yan, and W. Yan, "A method to speed up VLSI hierarchical physical design in floorplanning," in *Proceedings of International Conference on ASIC (ASICON)*, Guiyang, China, 2017, pp. 347–350.
- [2] R. Rajine Swetha, B. S. Babu, and K. Sumithra Devi, "A survey of various algorithms for VLSI physical design," *World Academy of Science, Engineering and Technology*, vol. 5, pp. 390–395, 2011.
- [3] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 4, pp. 748–761, 2021.
- [4] 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.
- [5] Y. Shi, K. Xue, S. Lei, and C. Qian, "Macro placement by wire-mask-guided black-box optimization," in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, New Orleans, LA, 2023, pp. 6825–6843.
- [6] Y. Lai, Y. Mu, and P. Luo, "Maskplace: Fast chip placement via reinforced visual representation learning," in *Advances in Neural Information Processing Systems 35 (NeurIPS)*, New Orleans, LA, 2022, pp. 24019–24030.
- [7] K. Xue, R.-T. Chen, X. Lin, Y. Shi, S. Kai, S. Xu, and C. Qian, "Reinforcement learning policy as macro regulator rather than macro placer," in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, Vancouver, Canada, 2024, pp. 140565–140588.
- [8] 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 *Proceedings of Design, Automation & Test in Europe Conference (DATE)*, Lyon, France, 2025, pp. 1–7.
- [9] M. Pan, N. Viswanathan, and C. C. N. Chu, "An efficient and effective detailed placement algorithm," in *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, 2005, pp. 48–55.
- [10] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *Global and Detailed Placement*, Cham, 2022, pp. 95–130.
- [11] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany, and D. Z. Pan, "ABCDPlace: Accelerated batch-based concurrent detailed placement on multithreaded cpus and gpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5083–5096, 2020.
- [12] Y. Lin, B. Yu, and D. Z. Pan, "Detailed placement in advanced technology nodes: A survey," in *Proceedings of IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Hangzhou, China, 2016, pp. 836–839.
- [13] I. Arvanitakis, G. K. Kranas, M. F. Dossis, A. Kakarountas, and A. N. Dadaliaris, "Assessing swapping policies as a detailed placement approach," in *Proceedings of South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, Piraeus, Greece, 2023, pp. 1–6.
- [14] W. Chow, J. Kuang, X. He, W. Cai, and E. F. Y. Young, "Cell density-driven detailed placement with displacement constraint," in *Proceedings of International Symposium on Physical Design (ISPD)*, Petaluma, CA, 2014, pp. 3–10.
- [15] 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.
- [16] M. Fogaça, G. Flach, J. Monteiro, M. O. Johann, and R. Reis, "Quadratic timing objectives for incremental timing-driven placement optimization," in *Proceedings of IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Monte Carlo, Monaco, 2016, pp. 620–623.
- [17] J. Vygen, "Algorithms for detailed placement of standard cells," in *Proceedings of Design, Automation & Test in Europe Conference (DATE)*, Paris, France, 1998, pp. 321–324.
- [18] A. Kahng, P. Tucker, and A. Zelikovsky, "Optimization of linear placements for wirelength minimization with free sites," in *Proceedings of Asia and South Pacific Design Automation Conference (ASPAC)*, Hong Kong, China, 1999, pp. 241–244.
- [19] A. B. Kahng, I. L. Markov, and S. Reda, "On legalization of row-based placements," in *Proceedings of ACM Great Lakes Symposium on VLSI (GLSVLSI)*, New York, NY, 2004, p. 214–219.
- [20] J. Monteiro, G. Flach, M. O. Johann, and J. L. A. Güntzel, "An analytical timing-driven algorithm for detailed placement," in *Proceedings of IEEE Latin American Symposium on Circuits & Systems (LASCAS)*, Montevideo, Uruguay, 2015, pp. 1–4.
- [21] D. U. Lim and H. Park, "Graph neural network-based detailed placement optimization framework," in *Proceedings of International Symposium on Quality Electronic Design (ISQED)*, San Francisco, CA, 2024, pp. 1–6.
- [22] D. Lim and H. Park, "Timing-driven detailed placement with unsuper-vised graph learning," in *Proceedings of Design, Automation & Test in Europe Conference (DATE)*, Lyon, France, 2025, pp. 1–7.
- [23] C. Georgakidis, I. Lilitis, G. Stanimeropoulos, and C. P. Sotiriou, "RAD-Place: A timing-aware radiation-hardening detailed placement scheme satisfying TMR spacing constraints," in *Proceedings of IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Athens, Greece, 2021, pp. 1–6.
- [24] V. S. Livramento, R. Netto, C. Guth, J. L. Güntzel, and L. C. V. dos Santos, "Clock-tree-aware incremental timing-driven placement," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 3, pp. 38:1–38:27, 2016.
- [25] T. Lee and Y. Li, "Incremental timing-driven placement with approximated signoff wire delay and regression-based cell delay," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 10, pp. 2434–2446, 2019.
- [26] S. Dutt, H. Ren, F. Yuan, and V. Suthar, "A network-flow approach to timing-driven incremental placement for asics," in *Proceedings of IEEE/ACM international conference on Computer-aided design (ICCAD)*, San Jose, CA, 2006, pp. 375–382.
- [27] J. A. S. Jesuthasan, "Incremental timing-driven placement with displacement constraint," Ph.D. dissertation, University of Waterloo, 2015.
- [28] M. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite," in *Proceedings of International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, 2015, pp. 921–926.
- [29] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design - From Graph Partitioning to Timing Closure*. Springer, 2011.
- [30] H. Ren, D. Z. Pan, C. J. Alpert, G.-J. Nam, and P. Villarubia, "Hippocrates: First-do-no-harm detailed placement," in *Proceedings of Asia and South Pacific Design Automation Conference (ASPAC)*, Yokohama, Japan, 2007, pp. 141–146.
- [31] W.-K. Fang and W.-K. Mak, "Placement flow study and detailed placement for hybrid-row-height designs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 6, p. 22, 2024.
- [32] Y.-Y. Lan and T.-C. Wang, "Detailed placement refinement for via pillar insertion and pin accessibility," in *Proceedings of International VLSI Symposium on Technology, Systems and Applications (VLSI TSA)*, Taiwan, China, 2024, pp. 1–4.
- [33] M. Naresh and S. Sachin, *Timing Analysis and Optimization of Sequential Circuits*. Springer, 1999.
- [34] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven placement," in *Handbook of Algorithms for Physical Design Automation*, 2008.
- [35] 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 International Symposium on Physical Design (ISPD)*, New York, NY, 2003, p. 60–66.
- [36] Z. Zhou, Y. Yu, and C. Qian, *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.
- [37] S. Jadav, S. Tayal, R. Chandel, and M. Vashishath, "High speed RLC equivalent RC delay model using normalized asymptotic function for global VLSI interconnects," *Microelectronics Journal*, vol. 107, p. 104941, 2021.
- [38] J. Cong, K.-S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed rc delay model," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, Dallas, Texas, 1993, pp. 606–611.
- [39] I. Ciofi, A. Contino, P. J. Roussel, R. Baert, V.-H. Vega-Gonzalez, K. Croes, M. Badaroglu, C. J. Wilson, P. Raghavan, A. Mercha, D. Verkest, G. Groeseneken, D. Mocuta, and A. Thean, "Impact of wire geometry on interconnect rc and circuit delay," *IEEE Transactions on Electron Devices*, vol. 63, no. 6, pp. 2488–2496, 2016.
- [40] T. Huang and M. D. F. Wong, "OpenTimer: A high-performance timing analysis tool," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Austin, TX, 2015, pp. 895–902.
- [41] Y. Shi, C. Gao, W. Ren, S. Xu, K. Xue, M. Yuan, C. Qian, and Z. Zhou, "Open3DBench: Open-source benchmark for 3D-IC backend implementation and PPA evaluation," *CoRR abs/2503.12946*, 2025.