Electrostatics-Based Analytical Global Placement for Timing Optimization*

Zhifeng Lin[1,2], Min Wei[2], Yilu Chen[1], Peng Zou[2], Jianli Chen[2], and Yao-Wen Chang[3]
[1]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China
[2]State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai 200433, China
[3]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
linzhifeng@fzu.edu.cn; chenjianli@fudan.edu.cn; ywchang@ntu.edu.tw

ABSTRACT

Placement is a critical stage for VLSI timing closure. A global placer without considering timing delay might lead to inferior solutions with timing violations. This paper proposes an electrostatics-based timing optimization method for VLSI global placement. Simulating the optimal buffering behavior, we first present an analytical delay model to calculate each connection delay accurately. Then, a timing-driven block distribution scheme is developed to optimize the critical path delay while considering the path-sharing effect. Finally, we develop a timing-aware precondition technique to speed up placement convergence without degrading timing quality. Experimental results on industrial benchmark suites show that our timing-driven placement algorithm outperforms a leading commercial tool by 6.7% worst negative slack (WNS) and 21.6% total negative slack (TNS).

1 INTRODUCTION

Placement is a critical stage in bridging logic design and physical layout. The performance of a placement engine has significant impacts on the overall area optimization, timing closure, and design turnaround time. Existing placement algorithms can be generally classified into three categories: simulated annealing, min-cut partitioning, and analytical formulation. Compared with the first two types of approaches, recent studies have shown that analytical placers can achieve the best placement result within reasonable runtime.

Typically, an analytical placement contains three major steps: global placement, legalization, and detailed placement. Global placement targets at producing nearly-legal layouts while globally optimizing the placement objective (*e.g.,* wirelength and routability). Legalization removes all block overlaps by locally perturbing the result of global placement. Detailed placement further conducts local block movement to improve the placement result. Among these steps, global placement dominates the final solution quality and thus attracts much attention from VLSI researchers [1].

For analytical global placement, most advanced placers (*e.g.,* the NTUplace family [2–4] and the ePlace family [5–7]) focus on wirelength and routability optimization, and formulate the wirelength- or routability-driven placement problem as nonlinear programming with respect to density constraints. As the design complexity keeps growing, circuit delay has become a crucial issue in VLSI placement. It is thus desirable to develop an effective timing-driven placement algorithm to address the crucial delay problem [8].

Timing-driven placement aims to find the desired block locations to improve the *worst negative slack* (WNS) and *total negative slack* (TNS). There are two types of timing-driven placement algorithms: path-based approaches and net-based approaches. Path-based approaches formulate the timing optimization problem as mathematical programming and explicitly minimize the delay of selected paths [9, 10]. Algorithms of this type have more accurate timing control but suffer from high time complexity. As a result, global placement often adopts net-based approaches to guide blocks to timing-friendly positions [11, 12]. This method can be easily incorporated into existing wirelength-driven placers without much runtime overhead. However, conventional net-based methods seldom consider the logic connection information (*i.e.,* pin-to-pin connections) in critical paths, which may produce inferior timing placement results.

In addition, the delay model in timing-driven placement is essential to timing optimization. Previous work [13] proved that delay is linear in terms of length in an optimal buffering, while the commonly used Elmore model [14] overlooks the delay characteristic and is quadratic in length. This mismatch in delay calculation leaves significant room for timing optimization, especially for the extremely long wires in modern large-scale designs.

In this work, we present an electrostatics-based analytical global placement algorithm for VLSI timing optimization. By leveraging the proposed linear delay model, we develop an effective timing-driven algorithm and adapt the placement results for subsequent physical design optimization. The major contributions of our work are summarized as follows.

- We develop a virtual buffer-based delay model to calculate each connection delay accurately. The delay model captures the optimal buffering behavior and is linear in terms of length.
- We present a new paradigm for timing-driven placement which optimizes the critical path delay while considering the path-sharing effect.
- We propose a timing-aware precondition technique to eliminate the gradient deviations induced by heterogeneous blocks and further speed up the timing convergence.
- Compared with the state-of-the-art academic placer and a leading commercial tool, the experimental results on industrial benchmark suites show that our analytical placer achieves the best timing result without degrading the routing overflows.

The remainder of this paper is organized as follows. Section 2 gives the preliminaries for timing-driven global placement. Section 3 presents our analytical placement algorithm and timing optimization strategy. Experimental results are reported in Section 4. Finally, Section 5 concludes this paper.

2 PRELIMINARIES

In this section, we first introduce the nonlinear placement method and then give the problem formulation of timing-driven global placement.

*This work was partially supported by the National Key Research and Development Program of China under Grant Numbers 2021YFA1003602 and 2022YFB4400500.

979-8-3503-4859-0/DATE24/© 2024 EDAA

2.1 Nonlinear Global Placement

The global placement problem is distributing circuit blocks into a fixed placement region. Traditional global placers usually divide the placement region into uniform bins and optimize the total half perimeter wirelength (HPWL) under the density constraints:

$$\begin{aligned} \min_{\mathbf{v}} \quad & W(\mathbf{v}) \\ \text{s.t.} \quad & P_b(\mathbf{v}) \leq M_b, \quad \text{for each bin } b, \end{aligned} \tag{1}$$

where $\mathbf{v} = (\mathbf{x}, \mathbf{y})$ is the block locations, $W(\mathbf{v})$ is the wirelength function, $P_b(\mathbf{v})$ is the potential function, and M_b is the target area in bin b. Since the HPWL is not differentiable, we adopt a weighted average (WA) model [15] for wirelength smoothing:

$$\hat{W}(\mathbf{v}) = \sum_{e \in E} \hat{W}_e(\mathbf{v}) = \sum_{e \in E} \left(\hat{W}_{e_x}(\mathbf{v}) + \hat{W}_{e_y}(\mathbf{v}) \right), \tag{2}$$

$$\hat{W}_{e_x}(\mathbf{v}) = \frac{\sum_{i \in e} x_i \exp\left(\frac{x_i}{\gamma}\right)}{\sum_{i \in e} \exp\left(\frac{x_i}{\gamma}\right)} - \frac{\sum_{i \in e} x_i \exp\left(\frac{-x_i}{\gamma}\right)}{\sum_{i \in e} \exp\left(\frac{-x_i}{\gamma}\right)}, \tag{3}$$

where E is the net set, and $\hat{W}_{e_x}(\mathbf{v})$ and $\hat{W}_{e_y}(\mathbf{v})$ are the WA wirelength model along the horizontal and vertical directions, respectively. In Equation (3), x_i gives the x-coordinate of block i and γ is a smoothing factor that controls the modeling accuracy. By using a penalty method, nonlinear global placers often convert Equation (1) into an unconstrained problem as follows:

$$\min_{\mathbf{v}} = \hat{W}(\mathbf{v}) + \lambda P(\mathbf{v}), \tag{4}$$

where λ is the density penalty factor. Consequently, gradient-based methods are employed to solve this unconstrained optimization problem.

2.2 Timing Optimization

To improve circuit reliability, modern placers shall consider timing performance during the placement process. WNS and TNS are two basic metrics in the timing domain. WNS provides the timing information of the worst critical path, while TNS gives the overall timing results of multiple critical paths. Typically, the problem formulation for timing-driven global placement can be stated as follows:

- **Timing-Driven Global Placement:** Given a placement layout and a set of blocks, determine the physical locations for all movable blocks such that the timing metrics (*e.g.,* WNS and TNS) are optimized.

Different from the wirelength that evaluates the local feature of a single net, timing metrics give the global characteristics of the circuit. Thus placement algorithms often invoke an external static timing analysis (STA) engine to address the timing optimization. The basic idea is to find target nets on timing-critical paths and assign higher weights to those nets. As a result, the objective of timing-driven global placement is to minimize the weighted wirelength:

$$\min_{\mathbf{v}} = \omega \hat{W}(\mathbf{v}) + \lambda P(\mathbf{v}), \tag{5}$$

where $\omega = \{\omega_1, \omega_2, ...\omega_{|E|}\}$ denotes the net weights.

3 OUR PLACEMENT ALGORITHM

The overall flow of our timing-driven global placement is summarized in Figure 1. It consists of three major stages: (1) electrostatics-based block distribution, (2) timing evaluation, and (3) timing optimization.

In the block distribution stage, we adopt the electrostatics-based method to spread blocks to their desired locations with minimized wirelength evenly. Then, by leveraging the proposed accurate delay model, timing evaluation gives the overall timing information of a circuit. After that, we reformulate the timing-driven placement problem and develop an effective virtual path-based net weighting strategy to optimize the circuit timing while preserving good placement convergence. Finally, the Abacus legalization method [16] is implemented to produce legalized placements.

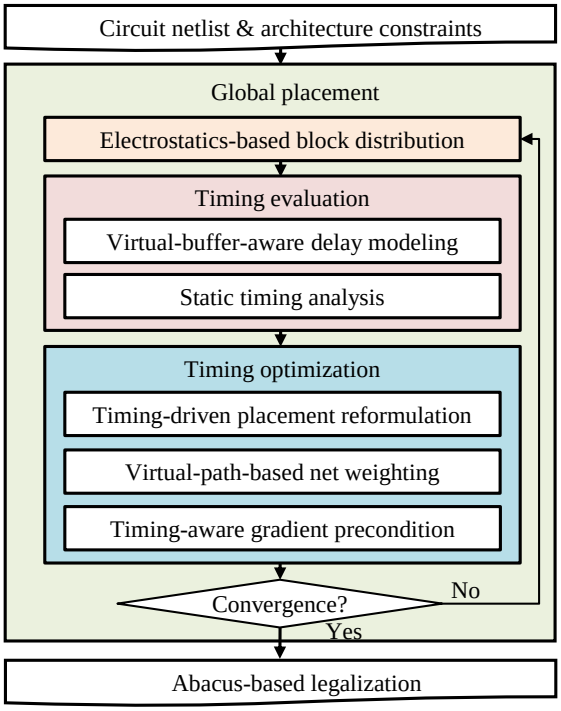

Figure 1: Our timing-driven global placement flow.

3.1 Electrostatics-Based Block Distribution

Placement engines need to distribute blocks to meet the density constraint evenly. Traditional nonlinear placers often smooth the density distribution by a bell-shape function [17] or a Sigmoid function [18]. In this paper, we employ an electrostatic-based density model [19] to promote block spreading without over sacrificing wirelength. In this model, each block i is treated as a charge with quantity q_i and potential energy N_i. Then, the density function P($\mathbf{v}$) in Equation (5) is converted to the total electric potential energy as follows:

$$N(\mathbf{v}) = \sum_{i \in V} N_i = \sum_{i \in V} \iint_{R_i} \psi_(x, y) dx dy, \tag{6}$$

where V is the block set, and $R_i = [x_i - \frac{w_i}{2}, x_i + \frac{w_i}{2}] \times [y_i - \frac{h_i}{2}, y_i + \frac{h_i}{2}]$. Here, w_i, and h_i represent the width and height of the block i, respectively. In Equation (6), $\psi(x, y)$ is the solution of the DC-removed Poisson equation introduced in [1].

By using the variable separation method, the work [20] obtains an analytical solution to the Poisson equation. The truncated version is as follows:

$$\psi(x, y) = \sum_{u=0}^{K} \sum_{p=0}^{K} a_{up} cos(\frac{u\pi}{W} x) cos(\frac{p\pi}{H} y), \tag{7}$$

where K is the truncation factor, a_{up} is the coefficient computed by the Fast Fourier Transform (FFT), and W and H are the width and height of the placement region, respectively. Substituting Equation (7) into Equation (6), we have

$$\begin{aligned}
N_i &= \iint_{R_i} \psi_(x,y)dxdy \\
&= \sum_{u=0}^{K}\sum_{p=0}^{K} a_{up} \int_{x_i-\frac{w_i}{2}}^{x_i+\frac{w_i}{2}} cos(\frac{u\pi}{W}x)dx \int_{y_i-\frac{h_i}{2}}^{y_i+\frac{h_i}{2}} cos(\frac{p\pi}{H}y)dy \\
&= \sum_{u=1}^{K}\sum_{p=1}^{K} a_{up}\frac{WH}{up\pi^2}\left[sin\frac{u\pi(x_i+\frac{w_i}{2})}{W} - sin\frac{u\pi(x_i-\frac{w_i}{2})}{W}\right] \\
&\quad \left[sin\frac{p\pi(y_i+\frac{h_i}{2})}{H} - sin\frac{p\pi(y_i-\frac{h_i}{2})}{H}\right] \\
&\quad + \sum_{u=1}^{K} a_{u,0}\frac{Wh_i}{u\pi}\left[sin\frac{u\pi(x_i+\frac{w_i}{2})}{W} - sin\frac{u\pi(x_i-\frac{w_i}{2})}{W}\right] \\
&\quad + \sum_{p=1}^{K} a_{0,p}\frac{Hw_i}{p\pi}\left[sin\frac{p\pi(y_i+\frac{h_i}{2})}{H} - sin\frac{p\pi(y_i-\frac{h_i}{2})}{H}\right].
\end{aligned} \tag{8}$$

Accordingly, we can get the gradient of the density function for the horizontal movement of block i as follows:

$$\begin{aligned}
\frac{\partial N_i}{\partial x_i} &= \sum_{u=1}^{K}\sum_{p=1}^{K} a_{up}\frac{H}{p\pi}\left[cos\frac{u\pi(x_i+\frac{w_i}{2})}{W} - cos\frac{u\pi(x_i-\frac{w_i}{2})}{W}\right] \\
&\quad \left[sin\frac{p\pi(y_i+\frac{h_i}{2})}{H} - sin\frac{p\pi(y_i-\frac{h_i}{2})}{H}\right] \\
&\quad + \sum_{p=1}^{K} a_{u,0}h_i\left[cos\frac{u\pi(x_i+\frac{w_i}{2})}{W} - cos\frac{u\pi(x_i-\frac{w_i}{2})}{W}\right].
\end{aligned} \tag{9}$$

The vertical gradient can be derived in the same manner. By leveraging the smoothed density gradient, we can distribute blocks to their desired locations with satisfied density constraints.

3.2 Virtual Buffer-Aware Delay Modeling

To achieve high-quality timing solutions, an accurate and effective delay model is required. Existing timing-driven global placers often adopt the Elmore function for net delay calculation. Given the distributed RC network of a net, the Elmore function can be derived as follows:

$$d_{s\to t} = \sum_{a\in V'} R_{a\to t}C_t, \tag{10}$$

where $d_{s\to t}$ is the Elmore delay from net source s to its sink t, V' is the node set in the RC network, C_t is the capacitance at node t, and $R_{a\to t}$ is the total resistance of the common path between the paths from s to a and s to t. In Equation (10), the resistance $R_{a\to t}$ and the capacitance C_t are linear to the wirelength, making the Elmore delay quadratic in length. However, in optimal buffering, the delay is linear in terms of length. Thus, the Elmore function may be too pessimistic about achieving the desired timing solution, especially for the extremely long wires in modern large-scale designs.

In this paper, we propose a virtual buffer-aware linear model to honor the delay characteristics and adapt the timing-driven placement solutions to the downstream physical optimization. Let r and c be the unit resistance and capacitance of a circuit, respectively. For a two-pin net with l length, we divide it into k segments by $k-1$ buffers. Suppose that the input capacitance of the net sink is equal to that of the buffer, and all of the buffers and the net driver have equal skews. Then, we have

$$\begin{aligned}
D_{cell} &= m_c L_{out} + b_{sc}(S_{in})^{m_{sc}} \\
&= m_c(\frac{cl}{k} + C_{in}) + b_{sc}(S_{in})^{m_{sc}},
\end{aligned} \tag{11}$$

where m_c, b_{sc}, and m_{sc} are constants, D_{cell} is the cell delay of the inserted buffer, L_{out} is the output capacitance, S_{in} is the input slew, and C_{in} is the input capacitance. Similarly, the output slew of each buffer can be approximated with

$$\begin{aligned}
S_{out} &= m_s L_{out} + m_{ss}(S_{in}) + b_{ss} \\
&= m_s(\frac{cl}{k} + C_{in}) + m_{ss}(S_{in}) + b_{ss},
\end{aligned} \tag{12}$$

where m_s, m_{ss}, and b_{ss} are constants.

After determining the buffer delay, the total delay of this two-pin net can be formulated as follows:

$$D_{net} = (k-1)D_{cell} + k\cdot\frac{rcl^2}{2k^2}. \tag{13}$$

Since all of the buffers have the same slew, S_{in} and S_{out} in Equation (12) should be the same. As a result, we have

$$S_{in} = \frac{m_s(\frac{cl}{k} + C_{in}) + b_{ss}}{1 - m_{ss}}. \tag{14}$$

Given that the variation of k affects the second term in Equation (11) much less than the first term, we assume that the second term in Equation (11) is constant. Substituting Equation (11) into Equation (13) and take derivative to zero, we have

$$\begin{aligned}
\frac{dD_{net}}{dk} &= m_c C_{in} + \frac{m_c cl}{k}^2 + b_{sc}(S_{in})^{m_{sc}} - \frac{rcl^2}{2k^2} = 0, \\
k &= \sqrt{\frac{\frac{rcl^2}{2} - m_c cl}{m_c C_{in} + b_{sc}(S_{in})^{m_{sc}}}}.
\end{aligned} \tag{15}$$

Combining Equations (13) and (15), we can obtain the accurate net delay with optimal buffer spacing, which follows the delay characteristic and is linear to the net length.

After obtaining the accurate net delay, we implement static timing analysis to evaluate the circuit timing performance. Specifically, we model the circuit timing as a directed acyclic graph, in which the input and output pins of a circuit are modeled as nodes, and the nets connecting pins are represented as edges. A STA engine computes the arrival time and required time of nodes through propagation, the resulting slack values are then used to guide timing optimization during the placement process.

3.3 Timing-Driven Placement Reformulation

As introduced in Section 2.2, timing-driven placers aim to distribute blocks to the desired locations with optimized timing results. For the critical blocks identified by static timing analysis, traditional timing-driven placement engines try to find the physical nets connecting to these critical blocks and then assign higher weights to those nets to optimize timing. However, in modern large-scale designs, a physical net often contains many other terminals except for critical ones. Assigning higher weight to such nets may not work well on the target critical blocks, leading to inferior timing solutions.

To address this issue, we propose to add virtual two-pin nets to link the blocks on the critical path. Then, higher attention is paid to these two-pin nets, which aids placement engines to minimize the two-pin net length effectively, thus optimizing the path timing results. In addition, for the movable blocks on the critical paths,

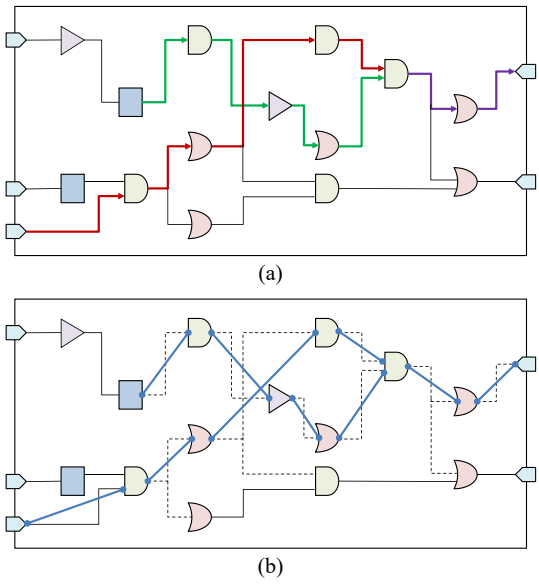

Figure 2: Illustrations of our path linking and output net enhancing methods. (a) An example netlist with two critical paths. (b) The critical path linking results. Solid lines give the added virtual two-pin nets, and dotted lines represent the extracted output nets.

their output nets are also assigned higher weights. By moving the blocks adjacent to the critical path in its fanout zone, we can further speed up the timing closure. Figure 2 illustrates our path linking and output net enhancing strategies. In Figure 2(a), there are two example critical paths represented by bold arrows. The common parts of these two paths are shown in blue. Figure 2(b) gives the extracted output nets (shown by dotted lines) and the corresponding path-linking results (blue solid lines connect the blocks on the critical path).

Based on our critical path linking and output net enhancing scheme, we reformulate the timing-driven placement problem in Equation (5) as follows:

$$\min_{\mathbf{v}} = \omega \hat{W}(\mathbf{v}) + \lambda N(\mathbf{v}) + \mu VL(\mathbf{v}), \tag{16}$$

where $N(\mathbf{v})$ is electrostatics-based density function described in Section 3.1, $VL(\mathbf{v})$ is the virtual net wirelength, $\mu = \mu_1, \mu_2, ...\mu_{|\hat{E}|}$ represents the virtual net weights, and $|\hat{E}|$ represents the set of added virtual net. To guarantee analyticity, we adopt the weighted average model described in Section 2.1 to smooth the virtual net wirelength. By using the smoothed gradient of the virtual net wirelength, we can place timing-critical blocks closer, leading to better timing solutions.

3.4 Virtual Path-Based Net Weighting

Timing-driven placers often assign weights to nets based on their timing criticality. Let s_{wns} be the worst negative slack of a circuit. The timing criticality of a specific net e can be modeled as follows:

$$c_e = \begin{cases} 0, & \text{if } s_{wns} \geq 0, \\ max\{0, \frac{s_e}{s_{wns}}\}, & \text{otherwise,} \end{cases} \tag{17}$$

where s_e is the slack of e. From Equation (17), if s_{wns} is non-negative, the corresponding timing performance is promising, and placement engines will do nothing to the net weights. Otherwise, the net criticality is given by the maximum value between 0 and s_e/s_{wns}. Typically, nets with larger criticality values are considered to be more sensitive to timing metrics, and placement algorithms would assign higher weights to them accordingly.

The criticality-based net weighting scheme is a popular mechanism in timing-driven global placement. Considering that an edge may be passed by a large number of critical paths in modern designs, net weighting algorithms considering this path-sharing effect would achieve better timing rewards. Based on the above discussion, we develop a novel virtual path-based weighting algorithm.

Algorithm 1 Virtual Path-Based Weighting Algorithm

Input: A timing graph tg,
Output: The updated netlist with weighted virtual nets.

```
1:  paths, slacks ← doTimingAnalysis(tg);
2:  for each path p in paths
3:    for each edge (s, t) in p
4:      if pins s and t belong to the same blocks
5:        continue;
6:      end if
7:      index ← the index of p;
8:      edge_slk[(s, t)].emplace_back(slacks[index]);
9:    end for
10: end for
11: for each entry [edge, slks] in edge_slk
12:   weight = getEdgeWeight(slks);
13:   addVirtualNet(edge, weight);
14: end for
```

Algorithm 1 details our efficient net weighting strategy. At line 1, we use the $doTimingAnalysis(tg)$ function to get a set of critical paths and their corresponding slacks. Then, during every iteration, we record the slack of the critical path passing through the edge (s, t) (lines 2–10). Finally, for each edge in the critical path, we use the $computeEdgeWeight(edge, slks)$ function to compute edge weights, and add a relevant weighted two-pin virtual net to the netlist. Specifically, the weight of each two-pin net is determined as follows:

$$Weight(s,t) = 1 + \sum_{(s,t)\in p} (R(S(p), T) - 1), \tag{18}$$

where

$$R(S(p), T) = \begin{cases} (1 - S(p)/T)^{\beta}, & \text{if } S(p) < 0, \\ 1, & \text{if } S(p) \geq 0. \end{cases} \tag{19}$$

With Equations (18) and (19), we scale the impact of all paths into timing criticality. Here, β is a user-defined parameter that controls the relative importance of paths with different criticalities. An edge passed by a large number of paths with high criticality values would be assigned a heavy weight. In addition, to improve the timing convergence, the weight for the output net of a movable block is increased by 0.5 × the weight of the virtual net started from this block. By leveraging the virtual path-based net weighting scheme, we can optimize the circuit delay effectively while improving the timing closure.

3.5 Timing-Aware Gradient Precondition

Due to the difference in dimension and connectivity between heterogeneous blocks, the gradient forces across different instances could have large deviations. Typically, compared with standard cells, macros are much larger in dimension and also have higher connectivity. The gradient deviations make the macros with larger movements

or even oscillations, causing the placement solution to fail to converge within a limited number of iterations.

Inspired by the advanced ePlace preconditioner [1], we propose a timing-aware precondition technique to address this gradient deviation problem while optimizing circuit timing. Different from traditional precondition methods that aim at solving the inverse Hessian matrix H_f^{-1} of the objective function f, our preconditioner approximates the Hessian matrix to a diagonal Jacobi matrix $\widetilde{H}_f$ to further reduce computational efforts. Subsequently, we multiply the gradient vector by the diagonal matrix and use the preconditioned gradients $\nabla f_{pre} = \widetilde{H}_f \nabla f$ to guide blocks to timing-friendly locations without suffering from gradient deviation.

Specifically, let $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ be the horizontal block locations, the i-th diagonal entry of the Jacobi matrix $\widetilde{H}_f$ can be derived as follows:

$$\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} = \omega \frac{\partial^2 \hat{W}(\mathbf{v})}{\partial x_i^2} + \lambda \frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} + \mu \frac{\partial^2 VL(\mathbf{v})}{\partial x_i^2}. \tag{20}$$

Since differentiating the wirelength function is computationally expensive, we approximate the first term $\frac{\partial^2 \hat{W}(\mathbf{v})}{\partial x_i^2}$ in Equation (20) as a simple binary. The term is set to 1 when the i-th block is incident to net e. As a result, we have

$$\omega \frac{\partial^2 \hat{W}(\mathbf{v})}{\partial x_i^2} = \sum_{e \in E_i} \omega_e \frac{\partial^2 \hat{W}(\mathbf{v})}{\partial x_i^2} \approx \sum_{e \in E_i} \omega_e, \tag{21}$$

where E_i is the net subset incident to block i and ω_e is the weight of net e. For the density term in Equation (20), we adopt the same preconditioning strategy proposed in [19] as follows:

$$\lambda \frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} \approx \lambda q_i. \tag{22}$$

The third term in Equation (20) is the virtual wirelength, which is employed to minimize the length of critical paths. Similar to Equation (21), we approximate the second-order derivative of the virtual wirelength function as the relevant net weights as follows:

$$\mu \frac{\partial^2 VL(\mathbf{v})}{\partial x_i^2} = \sum_{\hat{e} \in \hat{E}_i} \mu_{\hat{e}} \frac{\partial^2 VL(\mathbf{v})}{\partial x_i^2} \approx \sum_{\hat{e} \in \hat{E}_i} \mu_{\hat{e}}, \tag{23}$$

where $\hat{E}_i$ is the virtual net subset incident to block i and $\mu_{\hat{e}}$ is the weight of virtual net $\hat{e}$. After calculating all the gradients, we can formulate the preconditioning matrix in the horizontal direction as follows:

$$\widetilde{H}_{f_x} = diag\left(\sum_{e \in E_i} \omega_e + \lambda q_i + \sum_{\hat{e} \in \hat{E}_i} \mu_{\hat{e}}, ..., \sum_{e \in E_n} \omega_e + \lambda q_n + \sum_{\hat{e} \in \hat{E}_n} \mu_{\hat{e}} \right). \tag{24}$$

The preconditioning matrix in the vertical direction can be derived in the same manner. With this timing-aware precondition technique, we can eliminate the gradient deviation and speed up the timing closure.

4 EXPERIMENTAL RESULTS

We implemented our timing-driven global placement algorithm in the C++ programming language and conducted experiments on industrial benchmark suites. Table 1 gives the benchmark statistics, where "Period" represents the clock period in nanoseconds, and "#Blocks", "#Nets" and "#Pins" denote the number of blocks, nets, and pins, respectively. All the benchmarks are relatively large, and most of them contain millions of blocks and nets.

To study the performance of our placer, we first compared our analytical timing-driven placement strategy (named "Ours") with the state-of-the-art academic placer DREAMPlace [7]. Then, the advanced commercial tool Cadence Innovus 20.13 with the default settings (represented by "iTool") is used to further evaluate the effectiveness of our timing optimization algorithm. We ran all the experiments on the same Linux workstation with Intel Xeon 2.4GHz and 1TB memory and used the "iTool" to report the timing and routability results.

Table 1: Benchmark statistics

Benchmarks	Period	#Blocks	#Nets	#Pins
AIC_FSU	0.82	2927402	2551504	9528634
AHB_HIFEX_WRAP	1.25	1127904	1216577	4335230
LX930b_CLAMP_WRAP	0.34	918622	685764	2542951
ENYO_CORE_SUBSYS	0.4	2133417	1989797	7498861
SEC_CORE_8D6T	2.5	1427223	1461502	5612346
ROCKET_DLA_TOP	0.95	2048857	1824439	6810818
DDRC_HIFEX_WRAP	1.5	839581	857185	3254280
RSFEC_MTE_W	0.56	2157315	1984282	7301676
RX_DEC_SUBSYS	1.68	4843973	4797617	16959598
lx930b_SYSTEM	0.34	1977236	1968984	7372134
AHB_NOISE_SRC	1.25	1527566	1552657	5691639
CPU_VI_DEC_TOP	1	1021481	952813	3500770
DDRC_MO_M2_RING	0.63	725039	762019	2860185
HP_ISMO_TYPE1	1.5	1657690	1672977	6014245
MEDIA0_VI_SUBSYS	2.5	1270427	1435260	5245347

Table 2 lists the comparisons of the solution quality among "Ours", "DREAMPlace", and "iTool", in which "RT" gives the placement CPU time in seconds, and "WNS" and "TNS" represent the worst negative slack and the total negative slack in nanoseconds, respectively. In addition, the "HOF" and "VOF" in Table 2 denote the routing overflow ratios in the horizontal and vertical directions, respectively. Considering that the values of "HOF" and "VOF" are relatively small, we compared the average value instead of the average ratio. Besides, since the obtained WNS is close to 0, we employed the actual circuit period for better WNS evaluation:

$$Norm_{wns} = \frac{Period - Innovs_{wns}}{Period - Ours_{wns}}. \tag{25}$$

From Table 2, it can be seen that our analytical timing-driven placer gives the best timing result without degrading the routing overflows. Specifically, our algorithm achieves 23.7% WNS improvement and 40.3% TNS optimization over "DREAMPlace", yet with a reasonable runtime overhead of 13.5%. The improvement of the slack comes from the consideration of timing optimization in our analytical global placement. For the commercial tool "iTool", our placer outperforms it in WNS, TNS, and runtime. The most notable result on AHB_NOISE_SRC improves WNS and TNS by 33% and 41%, respectively. On average, our placement algorithm can achieve 6.7% WNS optimization and 21.6% TNS improvement within 15.3% shorter placement runtime. Besides, the placements produced by our algorithm also preserve good routability with only 0.086% degradation on vertical congestion ratios. The experimental results show that our placement algorithm is effective and efficient for timing-driven placement problems.

Finally, as shown in Figure 3, we report the runtime breakdown of our proposed algorithm based on the benchmark AHB_NOISE_SRC. It can be seen that the majority (60.58%) runtime is taken on the objective and gradient calculation, which is iteratively employed for solving the unconstrained placement problem by nonlinear programming.

Table 2: Comparison of WNS, TNS, HOF, VOF, and RT with DREAMPlace, iTool and our algorithm

Benchmark	DREAMPlace [7]					iTool					Ours				
	WNS	TNS	HOF	VOF	RT	WNS	TNS	HOF	VOF	RT	WNS	TNS	HOF	VOF	RT
AIC_FSU	-2.64	-5213	0.18	0.04	9058	-2.32	-4988	0.2	0.07	11438	-1.896	-3987	0.15	0.09	10179
AHB_HIFEX_WRAP	-0.35	-23	2.45	4.02	5309	-0.1	-19	2.17	4.08	6268	-0.07	-15	2.06	4.38	5807
LX930b_CLAMP_WRAP	-0.56	-1364	0.01	0.02	2330	-0.37	-1235	0.02	0.02	3023	-0.311	-1192	0.03	0.02	2612
ENYO_CORE_SUBSYS	-1.24	-375	0.1	0.08	5028	-0.87	-285	0.11	0.07	7713	-0.626	-271	0.05	0.1	5640
SEC_CORE_8D6T	-0.22	-12	2.14	1.64	4619	-0.09	-10	2.48	1.07	7304	-0.08	-8	2.23	1.62	5382
ROCKET_DLA_TOP	-0.43	-206	0	0	3861	-0.15	-185	0	0.01	5937	-0.157	-183	0	0	4326
DDRC_HIFEX_WRAP	-0.08	-33	0.11	0.28	3346	-0.07	-29	0.1	0.38	4857	-0.04	-24	0.14	0.31	3929
RSFEC_MTE_W	-0.22	-334	0	0	5919	-0.11	-285	0	0.01	9602	-0.14	-270	0	0.02	7183
RX_DEC_SUBSYS	-0.74	-1324	0.05	0.06	29066	-0.36	-991	0.19	0.04	31104	-0.287	-818	0.11	0.02	32134
lx930b_SYSTEM	-0.33	-2334	0.01	0	4606	-0.24	-1861	0	0	6901	-0.204	-1303	0	0.01	5338
AHB_NOISE_SRC	-0.95	-57	0.98	3.43	7484	-0.6	-55	0.84	3.13	9131	-0.14	-39	1.05	3.42	8896
CPU_VI_DEC_TOP	-0.16	-72	0.03	0.18	3608	-0.08	-68	0.01	0.18	4261	-0.04	-52	0.01	0.21	4416
DDRC_MO_M2_RING	-0.32	-13	1.43	3.54	3141	-0.16	-11	0.95	3.82	3708	-0.13	-10	1.59	3.36	3629
HP_ISMO_TYPE1	-0.43	-54	0.54	5.12	7254	-0.16	-49	0.73	5.71	8499	-0.07	-38	0.45	5.86	8263
MEDIA0_VI_SUBSYS	-0.21	-82	0.54	2.21	6423	-0.09	-76	0.48	2.13	7798	-0.2	-56	0.28	2.59	8103
Norm.Average	1.237	1.403	0.571	1.375	0.865	1.067	1.216	0.552	1.381	1.153	1.000	1.000	0.543	1.467	1.000

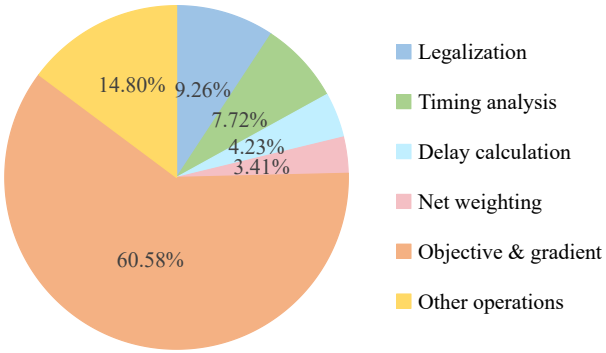

Figure 3: Runtime breakdown of the benchmark AHB_NOISE_SRC.

5 CONCLUSIONS

In this paper, we have proposed an electrostatics-based global placement algorithm for timing optimization. By simulating the optimal buffering behavior, we have first presented an analytical delay model to compute each connection delay accurately. Then, a new timing-driven global placement scheme has been developed to optimize the circuit timing while considering the path-sharing effect. Finally, we have proposed a timing-based precondition technique to eliminate gradient deviation and improve the placement convergence further. The experimental results on industrial benchmark suites have shown that our algorithm outperforms a leading commercial tool in both worst negative slack and total negative slack.

REFERENCES

[1] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. ePlace: Electrostatics based placement using fast fourier transform and Nesterov's method. *ACM Transactions on Design Automation of Electronic Systems*, 20(2):17:1–17:34, 2015.

[2] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1228–1240, 2008.

[3] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang. NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1914–1927, 2014.

[4] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany. NTUplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(3):669–681, 2018.

[5] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng. ePlace-MS: Electrostatics-based placement for mixed-size circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(5):685–698, 2015.

[6] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang. RePlAce: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9):1717–1730, 2019.

[7] 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*, 40(4):748–761, 2021.

[8] Z. Guo and Y. Lin. Differentiable-timing-driven global placement. In *Proceedings of the IEEE/ACM Design Automation Conference*, pages 1315–1320, 2022.

[9] J. Jung, G.-J. Nam, L. N. Reddy, I. H.-R. Jiang, and Y. Shin. OWARU: Free space-aware timing-driven incremental placement with critical path smoothing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(9):1825–1838, 2018.

[10] 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 IEEE/ACM International Symposium on Physical Design*, pages 73–80, 2016.

[11] 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 *Proceedings of the IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition*, pages 939–944, 2022.

[12] 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 IEEE/ACM International Symposium on Physical Design*, pages 141–148, 2015.

[13] Ralph H. J. M. Otten. Global wires: Harmful? In *Proceedings of the IEEE/ACM International Symposium on Physical Design*, pages 104–109, 1998.

[14] W. C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of applied physics*, 19(1):55–63, 1948.

[15] Y.-W. Chang M.-K. Hsu and V. Balabanov. TSV-aware analytical placement for 3d ic designs. In *Proceedings of IEEE/ACM Design Automation Conference*, pages 1–6, 2011.

[16] P. Spindler, U. Schlichtmann, and F. M. Johannes. Abacus: Fast legalization of standard cell circuits with minimal movement. In *Proceedings of the IEEE/ACM International Symposium on Physical Design*, pages 47–53, 2008.

[17] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. In *Proceedings of the IEEE/ACM International Symposium on Physical Design*, pages 18–25, 2004.

[18] T.-H. Lin M.-K. Hsu, S. Chou and Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit designs. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 1–6, 2011.

[19] X. Li, K. Peng, F. Huang, and W. Zhu. PeF: Poisson's equation based large-scale fixed-outline floorplanning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Early Access, 2022.

[20] W. Zhu, Z. Huang, J. Chen, and Y.-W. Chang. Analytical solution of poisson's equation and its application to vlsi global placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 1–8, 2018.