Adaptive Net and Path Weighting for Timing-Driven Placement

Shijian Chen[1,2] , **Ye Cai**[1] , **Biwei Xie**[4,2] , **Xingquan Li**[3,2]

[1]College of Computer Science and Software Engineering, Shenzhen University
[2]Peng Cheng Laboratory
[3]School of Mathematics and Statistics, Minnan Normal University
[4]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences

chenshijian2019@email.szu.edu.cn; caiye@szu.edu.cn; xiebiwei@ict.ac.cn; fzulxq@gmail.com

Abstract

For the modern very-large-scale integrated (VLSI) circuit design, placement plays an important role in timing closure and routability. To achieve a placement with better timing result, this paper designs a timing-driven placement framework by introducing a novel incremental weighting mechanism. During placement iteration, we propose a net weight and path weight mechanism by evaluating the criticality of timing metrics to guide the direction and step of cell movement. To update the net and path weight procedure, we introduce a timing criticality graph, and use a momentum method to search next net weight. Compared with our incremental weighting mechanism, the state-of-the-art placer "ePlace" is 62.5% and 72.0% worse on timing metric total negative slack (TNS) and worst negative slack (WNS) respectively in ICCAD-2015 benchmarks.

1 Introduction

In VLSI design, placement is performed in the early stage of physical design, and placement almost decide the location of standard cells and the length of interconnects in the layout. Due to its complexity, the placement problem is generally divided into three steps, global placement (GP), legalization (LG), detail placement (DP) respectively. GP roughly spreads out standard cells, and these locations are corrected by LG&DP. With the increasing impact of interconnects in modern chip design, timing closure becomes extreme difficult in the post physical design [Lu, 2017]. The typical goal of placement is to minimize the total wirelength of a design as it indirectly affects the routability, power consumption, and timing of circuit. Although minimizing the total wirelength can improve timing in general, it inevitably ignores some of the more critical paths such as timing-critical paths [Shahsavani and Pedram, 2020]. Timing-driven placement aims at considering timing metric (ex. data slack) into cell placement. Since accurate timing information cannot be evaluated until post-routing stages, most timing-driven placers [Bock *et al.*, 2015] use Flute [Chu and Wong, 2007] as the Steiner tree generator to evaluate wire delay for timing analysis. And lots of work [Mangiras *et al.*, 2019] invoke the static timing analysis (STA) engine to evaluate timing metrics. But there are some works [Guth *et al.*, 2015] directly calculate the timing metrics according to Elmore model,

1.1 Previous Works

Current the state-of-the-art placers are analytical method. They formulate the global placement problem into an unconstrained optimization problem and solve it by gradient descent algorithms. ePlace [Lu *et al.*, 2015] is the outstanding representative of analytical placement. ePlace develops a novel placement density function "eDensity", using the first-order optimization method Nesterov to iteratively solve the placement problem. Based on ePlace, lots of workS are proposed to further optimize the density function [Cheng *et al.*, 2018] [Zhu *et al.*, 2018], runtime [Lin *et al.*, 2020] etc.

Timing-driven placement can be summarized in two directions, net-based approach and path-based approach respectively. Net-based approach converts the obtained timing metrics into net weights and involve them during the iterative process of placement [Liao *et al.*, 2022]. Due to its simplicity and effectiveness, net-Based approaches are often used in global placement stage. Path-based approach constrains timing critical paths to reduce critical timing violations. They transform the problem into a mathematical programming problem [Mangiras *et al.*, 2019], and solve the problem by some heuristic approaches [Guth *et al.*, 2015].

1.2 Our Contributions

To summarize previous timing-driven placement work, they lack holistic solutions. Timing is usually not considered in the iterative process of analytical placers. Besides, the placement work dedicated to improving timing focus on local correction of standard cell positions in detail placement stage, limiting the global movement of standard cells.

To solve these problems, we propose an adaptive net and path weighting for timing-driven placement framework. Our main contributions are summarized as follows:

- With timing metrics, we propose a flexible timing criticality evaluation. To achieve suitable timing accumulation, we present two timing criticality propagation schemes (max and sum) to adapt calculating WNS and TNS.

- With timing criticality, a timing momentum scheme is proposed to update the variation of net weight instead of net weight. Besides current step, this scheme also considers history step into update procedure, which can avoid the violent oscillation of net weights.
- To convert timing criticality into path weights, we present a offset search window to guide cell relocation.
- Our approach achieves 62.5% and 72.0% better on WNS and TNS respectively in ICCAD-2015 benchmarks compared with the state-of-the-art placer.

The remainder of this paper is organized as follows. Section 2 is the preliminary. We introduce timing criticality in Section 3. Section 4 presents the proposed weight method for net and path. Experimental results of our method are presented in Sections 5, respectively.

2 Preliminaries

In this section, we introduce the placement problem at first. Then we introduce timing metrics, and we formulate the timing-driven placement by net-weighting. Finally, we formulate the timing optimization problem.

2.1 Placement Problem

In VLSI placement problem, we treat the cells in a circuit as vertices $\mathcal{C}$, and the nets as hyperedges $\mathcal{N}$. Then we get the hypergraph $G(\mathcal{C},\mathcal{N})$. The basic goal of placement is to minimize the total wirelength, while satisfying the constraints of the circuit (density, timing, routability etc.)

$$\begin{aligned} \min_{x,y} \ &: \ \sum_{e\in\mathcal{N}} WL_e(x,y) \qquad (1)\\ \text{s.t.} \ &: \ D_b(x,y) \le D_0, \qquad \forall b \in \mathcal{B},\\ &: \ Other\ constraints. \end{aligned}$$

Here, (x,y) is the location of cell. $WL_e(x,y)$ represents wirelength. The layout is evenly divided into bins $\mathcal{B}$, and we calculate the density of each bin. $D_b(x,y)$ is density expression and D_0 is target density. Global placement usually considers wirelength and density. The density constraint is usually transformed as a penalty term into objective function.

$$\min_{x,y} \ : \ \sum_{e\in\mathcal{N}} WL_e(x,y) + \alpha \sum_{b\in\mathcal{B}} D_b(x,y), \qquad (2)$$

where α is the density penalty parameter, and $WL_e(x,y)$ can be modeled using the half-perimeter wirelength (HPWL),

$$WL_e^{HPWL}(x,y) = \max_{i\in e} x_i - \min_{i\in e} x_i + \max_{i\in e} y_i - \min_{i\in e} y_i. \quad (3)$$

Problem (2) can be solved by gradient optimization approach, if the wirelength and density objective function are smooth. A suitable approximation smooth function for HPWL is weighted average (WA) [Lu *et al.*, 2015].

$$\begin{aligned} WL_e^{WA}(x,y) = &\frac{\sum_{i\in e} x_i \exp(x_i/\gamma)}{\sum_{i\in e} \exp(x_i/\gamma)} - \frac{\sum_{i\in e} x_i \exp(-x_i/\gamma)}{\sum_{i\in e} \exp(-x_i/\gamma)} \qquad (4)\\ &+ \frac{\sum_{i\in e} y_i \exp(y_i/\gamma)}{\sum_{i\in e} \exp(y_i/\gamma)} - \frac{\sum_{i\in e} y_i \exp(-y_i/\gamma)}{\sum_{i\in e} \exp(-y_i/\gamma)}, \end{aligned}$$

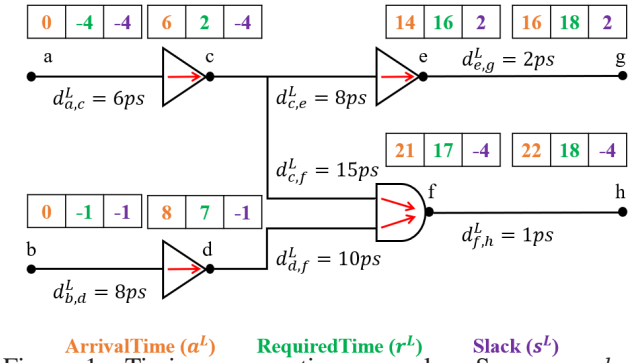

Figure 1: Timing propagation example. Suppose $a, b \in PI$ $g, h \in PO$ and required time in PO as 18. After timing propagation, we can calculate timing metrics and worst path $(a \to c \to f \to h)$.

where γ is the smoothing parameter that controls the accuracy of the model. ePlace [Lu *et al.*, 2015] proposes a novel density representations, called electric field method, which treats the cells as positive charges. The electric field method uses the Poisson's equation to model electric potential and electric field distribution, combined with Neumann boundary and compatibility conditions, to construct partial differential equations.

$$\begin{cases} \nabla\cdot\nabla\psi(x,y) = -\rho(x,y), \\ \hat{n}\cdot\nabla(x,y) = 0, \quad (x,y)\in\partial R, \\ \iint_R \rho(x,y)dxdy = \iint_R \nabla(x,y)dxdy, \end{cases} \qquad (5)$$

where, $\rho(x,y)$ is the electric density expression, which is using to described cell density. $\psi(x,y)$ represents the electric potential, and R is the placement region.

Finally, global placer calculates the gradients of wirelength and density ∇WL_e^{WA} and $\nabla\rho(x,y)$, and updates the location of cells by nesterov gradient method [Nesterov, 1983].

2.2 Timing Metrics

A sequential circuit can be represented by a set $\mathcal{C}$ of standard cells, set PI of primary inputs (circuit inputs or outputs of sequential cells) and a set PO of primary outputs (circuit outputs or inputs of sequential cells) [Guth *et al.*, 2015]. Timing violation can be classified as two case, the early case and the late case respectively [Kahng *et al.*, 2011].

Arrival times are measured at the output pin of each $c_j \in \mathcal{C}$ and at each $c_j \in (PI \cup PO)$. Let $\mathcal{F}_j$ be the set of cells connected to the input of $c_j \in \mathcal{C}$. Let $d^L_{i,j}$ and $d^E_{i,j}$ denote the late and early values for the delay measured between c_i's output and c_j's output. For each cell c_j, late arrival time a^L_j and early arrival time a^E_j are defined as below:

$$a^L_j = \max_{c_i\in\mathcal{F}_j}(a^L_i + d^L_{i,j}), \qquad a^E_j = \min_{c_i\in\mathcal{F}_j}(a^E_i + d^E_{i,j}). \quad (6)$$

Required time describes the signal which needs to arrive at the specified time, otherwise the clock frequency cannot be met [Albrecht *et al.*, 2002]. For each $c_j \in PO$, let r^L_j and r^E_j denote the late and early required time. To describe the extent of the timing violation, slack s^L_j and s^E_j are used to evaluate the difference between the arrival time and the required time.

$$s^L_j = r^L_j - a^L_j, \qquad s^E_j = a^E_j - r^E_j, \ \forall j \in PO. \quad (7)$$

The total negtive slack (TNS) and the worst negtive slack (WNS) describe the overall timing violation of a circuit.

$$TNS^{L/E} = \sum_{j \in PO} \min(0, slack_j^{L/E}), \ \forall j \in PO, \tag{8}$$

$$WNS^{L/E} = \min_{j \in PO} slack_j^{L/E}, \ \forall j \in PO. \tag{9}$$

In this paper, we resolve late violations first in the placement stage, since early violations are easier to resolve later. STA propagates arrival times forward from PI to PO in the reverse topological order [Hu *et al.*, 2015]. Figure 1 explains the calculation process of timing metrics between combinatorial logic cells.

2.3 Net Weighting Based Timing-Driven Global Placement

Net weighting is an effective way to distinguish the importance of connecting nets. Since it does not directly optimize timing metrics, it is easier to integrate net weights into existed analytical placement solvers.

However, timing violation information usually is obtained from STA tools in placement stage. STA tools use topology propagation to calculate overall timing metrics, while placement calculations are localized to cells or nets. In order to make up the incompatibility between the placer and the STA, a effective method is net weighting. With net weighting, Problem (2) can be formulated as follow Problem (10):

$$\min_{x,y} \ : \ \sum_{e \in \mathcal{N}} \omega_e \cdot WL_e(x,y) + \alpha \sum_{b \in \mathcal{B}} D_b(x,y), \tag{10}$$

where ω_e is timing net weight.

2.4 Timing Optimization Problem

Given the initial location (x_j^0, y_j^0) for standard cell j determined by global placement, the timing optimization problem is to find a new location (x, y) for each movable cell so as to minimize the timing violations.

$$\min \ : \ -\sum_{j \in PO} \hat{s}_j^L \tag{11}$$

$$\text{s.t.} \ : \ \hat{s}_j^L \leq 0, \ \forall j \in PO; \tag{12}$$

$$: \ r_j^L - a_j^L \geq \hat{s}_j^L, \ \forall j \in PO; \tag{13}$$

$$: \ a_i^L + d_{i,j}^L \leq a_j^L, \ \forall j \in \mathcal{C}. \tag{14}$$

Inequation (12) ensures that the selected slack values are all negative, avoiding paths with no timing violations. Inequation (13) states that negative slack is less than or equal to slack. And (14) defines late arrival time for each cell.

Similar to work in [Guth *et al.*, 2015], we relax the timing constraints by a non-negative Lagrange multiplier λ^L and incorporate them into the objective. With the Karush-Kuhn-Tucker (KKT) conditions and flow conservation, we can finally get the simplified function:

$$\min \ : \ \sum_{j \in \mathcal{C}} \sum_{i \in \mathcal{F}_j} \lambda_{i,j}^L \cdot d_{i,j}^L \tag{15}$$

where $\lambda_{i,j}^L$ is the path-weight between cells i and j. The function shows that reducing the product of the delay on path and the path-weight can also minimize $-\sum_{j \in PO} \hat{s}_j^L$, which can reduce timing violations.

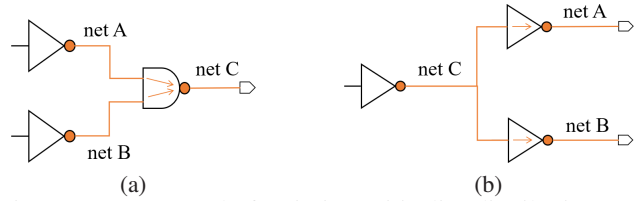

Figure 2: An example for timing criticality distribution. (a) Timing criticality distributes through cell. (b) Timing criticality distributes through net.

3 Timing Criticality

Timing criticality is used to indicate the importance of a net or path for timing. Subsequent cell movement processes need to be evaluated by timing criticality.

3.1 Timing Path Endpoint Criticality

As mentioned above, timing constraints are define in PO, so PO is the starting point for timing critical analysis. Equation (7) shows the relationship between slack and arrival time, required time. PO is timing path endpoint, when it's $slack < 0$, timing violation occurs. For each $j \in PO$, we use arrival time and required time to model the criticality of timing endpoints.

$$\mu_j^{k+1} = \mu_j^k \cdot (\frac{a_j^L}{r_j^L})^\beta, \ \forall j \in PO, \tag{16}$$

where μ_j^k represents the criticality of the k-th iteration of timing endpoint j. All initial timing criticality in PO are defined as 1. When timing violation occurs, the criticality value will be greater than 1. The more serious the violation, the larger the value, and the higher the corresponding criticality. The power term β controls the impact of amplifying timing violations, directing the optimizer to prioritize paths with more severe violations. In Equation (16), we update $_k$ by multiplication instead of addition, which is to keep the timing-critical information of the previous iteration and avoid timing-critical oscillations. For example, after violation being eliminated, the next iteration will regenerate the timing violation. Besides, the reason for using the ratio of arrival time and required time instead of directly adopting the slack value is that the slack value is fixed, and the timing critical value can be flexibly controlled in the form of a ratio.

3.2 Timing Criticality Propagation

After defining timing criticality of timing endpoint, the criticality of timing endpoint needs to be propagated. To calculate the timing criticality of each pin in a sequential circuit, the propagation mode is in reverse topological order.

The timing criticality distribution through cells is defined as followed :

$$\mu_{net_i,j} = \mu_{net_i,j} \cdot (\frac{a_i^L + d_{i,j}^L}{a_j^L}), \tag{17}$$

where $\mu_{net_i,j}$ represents timing criticality value from pin i to fanout cell outpin j. An example is shown in Figure 2(a). Suppose the critical value of net C comes from PO, and the nets on both sides of the cell should have corresponding criticality, so the critical value needs to be propagated. The timing

criticality proportion of net A from net C depends on whether it is the path with the maximum delay. Equation (17) is multiplied by a representative of historical information $\mu_{net_i,j}$, also to prevent timing oscillation. This paper proposes two propagation methods to control the degree of timing violations: a) max timing criticality method, and b) sum timing criticality method, respectively.

The max timing criticality method can find the worst timing path. We propose the max timing criticality propagation rule as follow:

$$\mu_{net_i} = \max_{j \in net_i} \mu_{net_i,j}, \tag{18}$$

where $\mu_{net_i,j}$ is timing criticality of a sink pin j of net i. As shown in Figure 2(b), net A and net B need to propagate timing criticality value to net C. The method of max timing criticality propagation is to select the value with the largest timing critical value among the two sink pins as the timing criticality of net C. Weighting high criticality of the net through max timing criticality propagation can effectively improve the worst path.

The sum timing criticality propagation can mark the most important pin among timing paths. we propose the sum timing criticality propagation rule as below :

$$\mu_{net_i} = \sum_{j \in net_i} \mu_{net_i,j}. \tag{19}$$

As shown in Figure 2(b), the sum of the timing criticalities of net A and net B assign to net C. By sum timing criticality propagation, timing criticality converges on the important pins that affect multiple timing paths [Huang and Wong, 2015].

In this paper, we manage these two propagation methods in a unified manner, and inprove TNS or WNS as needed.

4 Timing Driven Placement

4.1 Framework

In Figure 3, we summarize our timing-driven placement framework flow based on net-Weight and path-Weight mechanism. Gradient optimization iteratively updates cell location at global placement. During global placement process, net-weighting module evaluates the criticality of nets and updates the weight of nets. Then these nets will participate in the next round of parsing optimizer iterations. And then, the path-weighting cell relocation module is activated to repair timing violation, the cell relocation will be performed until the goal of global placement converges. If it is difficult to converge, it will even return to the analytical optimizer for large-scale cell movement.

4.2 Net Weighting Cell Movement

Our goal is to convert timing criticality into net weights that really participate in the optimization process. In order to quickly improve the worst timing violation during global placement stage, we adopt the max timing criticality propagation method. We first normalize for timing criticality.

$$\hat{\mu}_{net_i} = \frac{\mu_{net_i}}{\mu_{max}}. \tag{20}$$

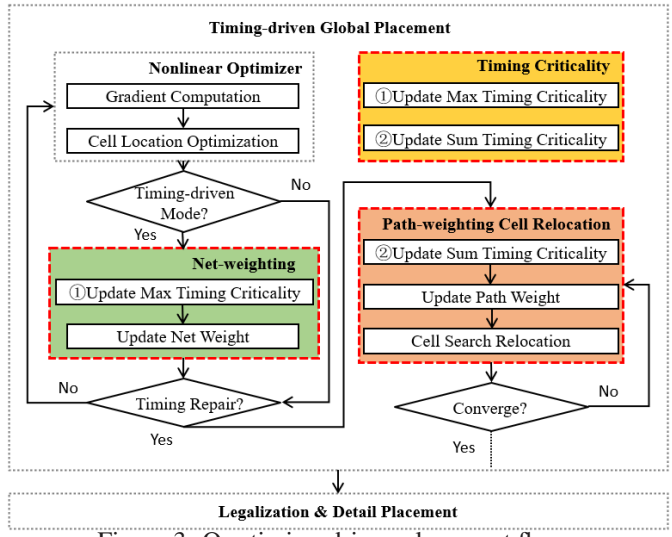

Figure 3: Our timing-driven placement flow

Algorithm 1 Net Weighting Based Cell Movement

Input: C : Circuit cell set , N : Circuit net set
Parameter: Density penalty α , Net wieght ω
Output: Optimal positions of C

```
for Convergence condition not reached do
    for all C_i ∈ C do
        WG_i ← Calculate wirelength gradient
        if timing-dirven mode then
            WG_i = ω · WG_i
        end if
        DG_i ← Calculate density gradient
        Gradient vector ∇F_i = WG_i + α · DG_i
    end for
    New position of C_i by gradient computation
end for
return Optimal positions of C
```

Using timing criticality directly as net weight will make some net weights close to zero without timing violations. Therefore, we apply the timing criticality to the increment of net weights. The advantage of setting the net weight increment is to only increase the weights of nets with timing violations, while keep the same net weight without timing violations.

Let $\hat{\mu}^k_{net_i}$ represent the timing criticality value of the net in the k-th round of iteration. Then we calculate the net weight increment in current iteration in combination with the criticality value of net in the $k-1$ round of iteration.

$$\Delta\omega^k_i = \theta \cdot \mu^{k-1}_i + (1-\theta) \cdot \mu^k_i, \forall i \in \mathcal{N}. \tag{21}$$

The momentum method is used to consider the net criticality in the previous step, and combines the net criticality in the current iteration. Then we can calculate new net weight incrementally. If net criticality value is large in previous step, and the value decrease sharply in current iteration, momentum method ensures that the net criticality value will not decrease sharply. The result is used as an increment to current net weight. For all nets, the increment decrease means that the net weight involved in the actual optimization process de-

crease. Finally, we update the net weight value.

$$\omega_i^k = \omega_i^{k-1} + \Delta\omega_i^k, \quad \forall i \in \mathcal{N}. \tag{22}$$

The net weights are integrated into the update process of the gradient optimization approach. In the iterative procedure, the increase in the wirelength gradient of a cell leads to an increase in the overall gradient of the cells. The procedure tends moving these cells to reduce the distance of the connected cells. During the global movement of cells driven by timing, the proximity of cells will lead to new density constraint violations. At this time, the penalty function term would be increased to ensure the spacing between cells.

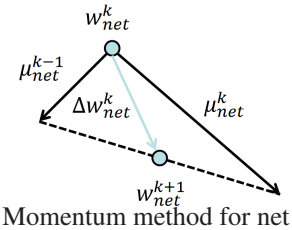

Figure 4: Momentum method for net weighting

4.3 Path Weighting Cell Relocation

Path-weighting can improve timing more accurately. The net contains multiple timing path segments, but some of them may be timing violation, and the rest would not. Therefore, path-weighting can label the importance of the timing path segments in net.

As mentioned in Equation (15), minimizing the product of path-weight and delay on path can effectively decrease timing violations. In this paper, path-weight is from the sum timing criticality propagation, and delay is between output pins of two connected cells. Since the location of cell will affect delay, we first analyze the direction of cell relocation. When late violation occurs at cell j, it should move to the direction with higher path weight to reduce path delay. As shown in Figure 4, the successor of cell C is cell D, and its predecessor cells are cell A and cell B. When cell C is in the timing violation path, vectors $v_{A,C}^L$, $v_{B,C}^L$, $v_{C,D}^L$ all point to the connected cell. The magnitude of each vector is equal to the value of timing criticality value between these two cells ($|v_{A,C}^L| = \mu_{A,C}^L$, $|v_{B,C}^L| = \mu_{B,C}^L$, $|v_{C,D}^L| = \mu_{C,D}^L$). The higher value of the timing criticality, which results in a "stronger" vector.

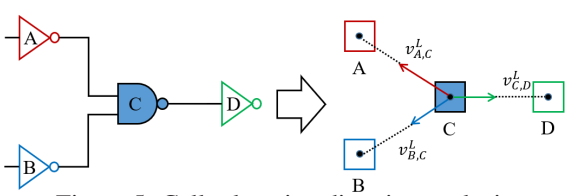

Figure 5: Cell relocation direction analysis

In this paper, the solution is to first determine that the cell is moved to a certain position. Then, we perform a winding evaluation on the cell at this position, and obtain the timing evaluation at this position to determine whether to move to a position favorable for timing. However, if the cell is moved in a large region, then the search space will be large and the convergence speed will be slow. Hence, it is necessary to set a suitable search window to reduce the search space.

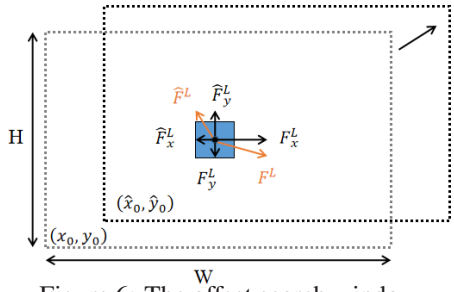

Figure 6: The offset search window

The offset of the search window depends on the path weights. The vector of cell connections is vectorized. As shown in Figure 6, the vector $F^L(\hat{F}^L)$ is decomposed into $F_x^L(\hat{F}_x^L)$ and $F_y^L(\hat{F}_y^L)$. The modulus of the vector is equal to the path weight. Suppose a cell i, its lower left corner coordinate is (x_i, y_i). All vector values in the positive direction of x are added to $|R_i|$, and all vector values in the negative direction of x are added to $|L_i|$. Similarly, we can get the positive direction of y to the y-axis $|U_i|$, and the negative direction of the y-axis $|D_i|$. We set search window size as $H * W$, and take the location of cell i as the center of window. The coordinate of the lower left corner of the search window is (x_0, y_0), where $x_0 = x_i - W/2$, $y_0 = y_i - H/2$. Finally, we obtain the location $(\hat{x}_0,\ \hat{y}_0)$ of the offset search window,

$$\hat{x}_0 = x_i - \frac{|L_i|}{|L_i| + |R_i|} \cdot W, \quad \hat{y}_0 = y_i - \frac{|D_i|}{|D_i| + |U_i|} \cdot H \tag{23}$$

Table 1: ICCAD 2015 contest benchmark statistics.

Benchmark	Cells	Nets	Pins	Rows
superblue1	1209716	1215710	3767494	1829
superblue3	1213253	1224979	3905321	1840
superblue4	795645	802513	2497940	1840
superblue5	1086888	1100825	3246878	2528
superblue7	1931639	1933945	6372094	3163
superblue10	1876103	1898119	5560506	3437
superblue16	981559	999902	3013268	1788
superblue18	768068	771542	2559143	1788

5 Experimental Results

We implemented our timing-driven placer in C++ programming language, and performed all experiments on Linux workstation with 2.3 GHz and 8 GB memory. The gradient optimization approach is based on open-source placer ePlace [Lu *et al.*, 2015]. We evaluated our timing-driven placer on large industrial designs from the ICCAD 2015 contest [Kim *et al.*, 2015]. Table 1 gives the benchmark statistics. A leading academic timer "UI-Timer" [Huang *et al.*, 2014] supported by the contest was used for timing analysis. We use "iTP-NetWeight" to represent the method in chapter 4.2,

Table 2: Comparison results on TNS, WNS and HPWL with "ePlace" and open source timing-driven placer "RePlAce-Timing".

Case	ePlace			RePlAce-Timing			iTP-NetWeight			iTP	
	TNS	WNS	HPWL	TNS	WNS	HPWL	TNS	WNS	HPWL	TNS	WNS
superblue1	-248.30	-25.89	79.63	-193.44	-22.31	79.64	-222.66	-20.84	**79.58**	**-185.97**	**-19.73**
superblue3	-72.01	-29.44	92.50	-49.58	-13.91	92.47	-38.91	-12.52	**92.24**	**-35.01**	**-11.64**
superblue4	-220.94	-25.92	**60.03**	**-182.68**	-14.50	60.06	-206.67	-13.20	60.08	-191.07	**-11.62**
superblue5	-190.08	-44.77	93.14	-154.57	-32.33	93.22	-150.73	-21.58	**92.89**	**-146.12**	**-20.46**
superblue7	-156.53	-17.45	113.58	-119.26	-15.21	113.57	-92.08	-15.21	**113.18**	**-78.52**	**-15.20**
superblue10	-660.60	-27.06	177.95	-588.40	-20.76	175.14	-558.40	-22.86	**174.69**	**-551.03**	**-20.53**
superblue16	-359.14	-36.72	85.22	-270.11	-28.27	**85.15**	-177.20	-19.07	85.35	**-154.40**	**-19.06**
superblue18	-76.32	-10.98	**46.96**	-51.67	**-8.80**	47.02	-53.49	-10.96	47.20	**-46.74**	-9.90
$Avg.Ratio^1$	×1.454	×1.611	×1.003	×1.111	×1.121	×1.001	**×1.000**	**×1.000**	**×1.000**	*	*
$Avg.Ratio^2$	×1.625	×1.720	*	×1.239	×1.192	*	1.113	1.068	*	**×1.000**	**×1.000**

TNS : in ($\times 10^5$) ps. **WNS** : in ($\times 10^3$) ps. **HPWL** : ($\times 10^6$) um. $Avg.Ratio^1$ on iTP-NetWeight. $Avg.Ratio^2$ on iTP.

and "iTP" is the full work of this paper. We compare "iTP-NetWeight" with the original "ePlace", and the open source timing-driven placer "RePlAce-Timing". Besides, we applied our timing-driven placer to the placement of an open source real chip "ysyx3", which includes 120,000 standard cells with 40M clock frequency.

In this paper, we use "FLUTE" [Chu and Wong, 2007] to obtain steiner trees for connected nets among standard cells [Chowdhary *et al.*, 2005]. Furthermore, global clock routing is assumed to be ideal (i.e., zero skew among Flip-Flop's clock inputs from clock source) to avoid producing a susceptible RC-tree topology for the clock network [Huang *et al.*, 2016].

We use timing metrics WNS and TNS , and HPWL to compare with other work. The metrics like overflow and runtime are not listed. Overflow is used as the termination condition of the algorithm, and the target condition can be reached by default. The main running time of our timing-driven placement is in the STA estimator part, because timing estimator is not belonging to the content of this paper. The solution in this paper focuses more on the solution quality, so it is not listed in the following index comparison. All timing criticality on the timing endpoint are initialized to 1, and the β is set to 2 when the timing violation occurs, otherwise 1. For the relocation of each cell, we identify 10 candidate positions uniformly spaced inside a rectangular search window of size $W = H = 20$ rows.

Table 2 shows overall TNS,WNS and HPWL comparison in net-weighting. These three works ("iTP-NetWeight", "ePlace" and "RePlAce-Timing") have the same global placement stage. The TNS and WNS of "ePlace" are 45.4% and 61.1% worse than iTP-NetWeight under the almost the same wirelength. On average, "RePlAce-Timing" is also 11.1% and 12.1% worse on TNS and WNS timing metrics.

iTP shows further timing inprovement through path-weighting cell relocation. In general, under the weight mechanism timing-driven global placement framework and algorithm proposed in this paper, Without considering timing, on timing metrics TNS, the analytical placer "ePlace" is 62.5% worse than iTP, and WNS is worse by 72.0%. For the "RePlAce-Timing" timing-driven placer, the overall timing metric TNS worse than iTP by 23.9% and WNS worse by 19.2%.

Table 3: Comparison results with "ePlace" in design "ysyx3".

Design	Tool	TNS	WNS	HPWL
ysyx3	ePlace	-85.9739	-393381	**70.859**
	iTP	**-81.4476**	**-370608**	71.959
Improvement Ratio(%)		+5.26%	+5.79%	-1.55%

TNS : in($\times 10^3$)ps. **WNS** : in($\times 10^3$)ps. **HPWL** : ($\times 10^5$)um.

We apply out timing-driven placer to the actual open source chip design "ysyx3". The result show that compared with "ePlace" without timing-driven, the TNS is improved by 5.26%, and WNS is improved by 5.79%. Although HPWL loses 1.55%, it is still within the acceptable range. The experimental results show that our weighting method is still effective for practical chip design.

6 Conclusion

In this paper, we propose a unified timing improvement framework that comprehensively considers path-weights and net-weights. By evaluating timing criticality and moving cells, we achieve great improvement on the worst negtive slack (WNS) and the total negtive slack (TNS), respectively on ICCAD-2015 benchmarks compared with the state-of-the-art placers. In addition, comparison on the actual open source design also verify the effectiveness of our placer. The future work will focus on optimizing skew between sequential cells to comprehensively consider timing in placement stage.

Acknowledgment

This work was supported in part by the Major Key Project of PCL (No. PCL2021A08), the National Natural Science Foundation of China (No. 61907024), the Natural Science Foundation of Fujian Province (No. 2020J05161), the Shenzhen Fundamental Research Program (No. 20200812112502001).

References

[Albrecht *et al.*, 2002] Christoph Albrecht, Bernhard Korte, Jürgen Schietke, and Jens Vygen. Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. *Discrete Applied Mathematics*, 123(1-3):103–127, 2002.

[Bock *et al.*, 2015] Adrian Bock, Stephan Held, Nicolas Kámmerling, and Ulrike Schorr. Local search algorithms for timing-driven placement under arbitrary delay models. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2015.

[Cheng *et al.*, 2018] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong 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, 2018.

[Chowdhary *et al.*, 2005] Amit Chowdhary, Karthik Rajagopal, Satish Venkatesan, Tung Cao, Vladimir Tiourin, Yegna Parasuram, and Bill Halpin. How accurately can we model timing in a placement engine? In *Proceedings of the 42nd annual Design Automation Conference*, pages 801–806, 2005.

[Chu and Wong, 2007] Chris Chu and Yiu-Chung Wong. Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83, 2007.

[Guth *et al.*, 2015] Chrystian Guth, Vinicius Livramento, Renan Netto, Renan Fonseca, José Luís Güntzel, and Luiz 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*, pages 141–148, 2015.

[Hu *et al.*, 2015] Jin Hu, Greg Schaeffer, and Vibhor Garg. Tau 2015 contest on incremental timing analysis. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 882–889. IEEE, 2015.

[Huang and Wong, 2015] Tsung-Wei Huang and Martin DF Wong. Opentimer: A high-performance timing analysis tool. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 895–902. IEEE, 2015.

[Huang *et al.*, 2014] Tsung-Wei Huang, Pei-Ci Wu, and Martin DF Wong. Ui-timer: An ultra-fast clock network pessimism removal algorithm. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 758–765. IEEE, 2014.

[Huang *et al.*, 2016] Chau-Chin Huang, Yen-Chun Liu, Yu-Sheng Lu, Yun-Chih Kuo, Yao-Wen Chang, and Sy-Yen Kuo. Timing-driven cell placement optimization for early slack histogram compression. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016.

[Kahng *et al.*, 2011] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.

[Kim *et al.*, 2015] Myung-Chul Kim, Jin Hu, Jiajia Li, and Natarajan Viswanathan. Iccad-2015 cad contest in incremental timing-driven placement and benchmark suite. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 921–926. IEEE, 2015.

[Liao *et al.*, 2022] Peiyu Liao, Siting Liu, Zhitang Chen, Wenlong Lv, Yibo Lin, and Bei Yu. Dreamplace 4.0: Timing-driven global placement with momentum-based net weighting. *Proc. DATE, Antwerp, Belgium*, 2022.

[Lin *et al.*, 2020] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David 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, 2020.

[Lu *et al.*, 2015] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. eplace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 20(2):1–34, 2015.

[Lu, 2017] Lee-Chung Lu. Physical design challenges and innovations to meet power, speed, and area scaling trend. In *Proceedings of the 2017 ACM on International Symposium on Physical Design*, pages 63–63, 2017.

[Mangiras *et al.*, 2019] Dimitrios Mangiras, Apostolos Stefanidis, Ioannis Seitanidis, Chrysostomos Nicopoulos, and Giorgos Dimitrakopoulos. Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2835–2848, 2019.

[Nesterov, 1983] Yu. E. Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In *Soviet Mathematics Doklady*, 1983.

[Shahsavani and Pedram, 2020] Soheil Nazar Shahsavani and Massoud Pedram. Tdp-admm: A timing driven placement approach for superconductive electronic circuits using alternating direction method of multipliers. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[Zhu *et al.*, 2018] Wenxing Zhu, Zhipeng Huang, Jianli Chen, and Yao-Wen Chang. Analytical solution of poisson's equation and its application to VLSI global placement. In *Proceedings of the International Conference on Computer-Aided Design*, pages 1–8, 2018.