Flexible Double-Side Pin Redistribution with Efficient Virtual-Net-Guided Cell Type Assignment

Xun Jiang[1], Haoran Lu[1], Yifan Chen[1], Yibo Zhang[1], Jianxiang Jin[1]
Jiarui Wang[1,2], Chunyuan Zhao[1], Heng Wu[1], Runsheng Wang[1,3,4], Yibo Lin[1,3,4*]
[1]School of IC, Peking University, [2]School of CS, Peking University, [3]Institute of EDA, Peking University,
[4]Beijing Advanced Innovation Center for Integrated Circuits, [*]Corresponding author: yibolin@pku.edu.cn

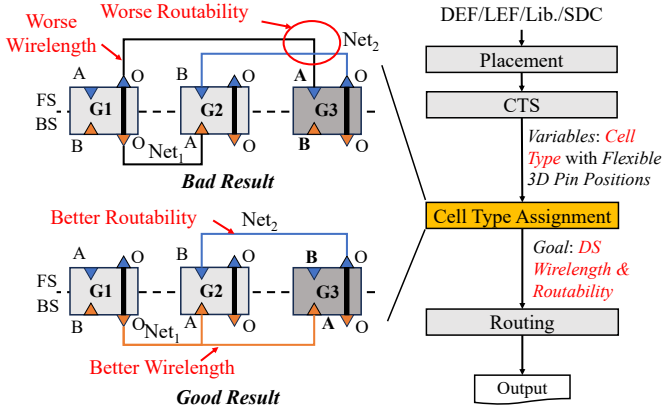
***Abstract*—The backside process has become a key enabling technology for sub-2nm nodes. To explore the design boundaries of double-side chips, leveraging cells with double-side pins is now at the forefront of design technology co-optimization (DTCO), whose key challenge is redistributing pins to the chip frontside or backside. In this work, we formulate this new problem mathematically and propose virtual-net-guided cell type assignment to solve it. Compared with the state-of-the-art (SOTA) method, our algorithm outperforms in all metrics and supports broader DTCO exploration. We believe our studies will largely stimulate research on double-side physical design and further push cutting-edge VLSI technology.**

I. Introduction

Backside (BS) interconnects and 3D-stacked transistors [1]–[5] are believed to be the key enabler for sub-2nm technology nodes and beyond [6], [7]. Both academia and industry are actively exploring the opportunities of double-side (DS) pins [8], [9] and interconnects [10]–[12] for further scaling. Intel has released its BS process solutions with six metal layers on the back of the wafer [6]. Meanwhile, TSMC has demonstrated its 3D-stacked CFET inverter solutions with multiple frontside (FS) and BS pin configurations [13].

The above emerging technologies bring new challenges in physical design. For example, each cell can be designed with types of different pin configurations, as a pin can be in the frontside, backside, or both sides [14], [15]. Improper cell type assignment can result in poor routability. As shown in the top of Fig. 1(a), if cell G3 chooses a type with G3/A on frontside and G3/B on the backside, then Net_1 has to detour from G2/A through backside G1/O, to frontside G1/O, and eventually be connected with G3/A. Such a scenario causes routability issues with Net_2. On the other hand, if cell G3 chooses a type with G3/A and G3/B swapped, as shown in the bottom of Fig. 1(a), both Net_1 and Net_2 can be connected locally with much better wirelength and routability. Such observation motivates the requirement of an additional cell type assignment (CTA) step for routability optimization in the design flow [14], [15], as a minor change in cell types cause drastic changes in wirelength and routability.

In literature, although several works [14]–[16] have investigated CTA, they are limited to very specific settings and simple strategies without holistic optimization of wirelength and routability, which are summarized in TABLE I. Firstly, these works [14]–[16] all adopt an identical setting of FS and BS metal layers, which lacks support for flexible FS/BS metal layer distributions. Lu et al. [14] aim to balance FS and BS pins by enforcing cell design with input pins distributed evenly on FS and BS. Choi et al. [16] cluster adjacent pins that are driven by the same nets or drive the same cells into several groups, which are assigned to frontside or backside. This method requires all the input pins of each cell on the same side. Ahn et al. [15] propose a two-step method: (1) assign cell type in the decreasing order of the total half-perimeter wirelength (HPWL) of nets connected to its input pins, and (2) flip all pins of a cell to the opposite side when congestion exists on one side. Step (2) also has to limit all the input pins of each cell on the same side. While this method achieves better results over [14], [16] due to direct optimization of wirelength and congestion, it is still limited to identical layer settings and cell types with all input pins on the same side.



(a) Comparison of CTA Results. (b) Flow [14], [15].

Fig. 1 In the DS physical design flow in (b), ***Bad Result*** and ***Good Result*** in (a) are affected by the cells colored in **dark gray**, whose input pins have different 3D positions.

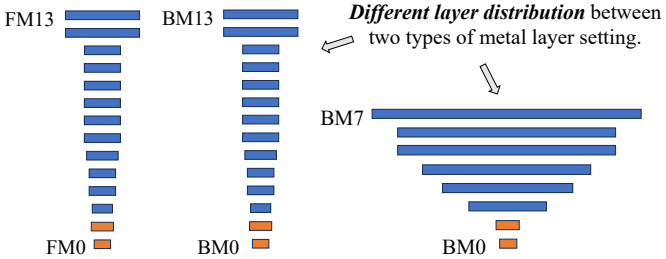
TABLE I A survey of related works on the CTA problem.

Work	Objective	FS/BS Layers	Cell Types
Lu et al. [14]	Balanced Pins	Identical	Not Flexible
Choi et al. [16]	Balanced Pins	Identical	Not Flexible
Ahn et al. [15]	WL & Cong.	Identical	Not Flexible
Ours	**WL & Cong.**	**Flexible**	**Flexible**

To achieve holistic optimization of wirelength and congestion under flexible layer configurations and cell types, in this work, we propose a virtual-net-guided cell type assignment method. We summarize the major contributions as follows:

- We propose a new general mathematical formulation for cell type assignment supporting various metal layer configurations and a complete set of cell types.
- We propose a new relaxation method to construct an efficient virtual net planning algorithm and further leverage virtual-net-guided cell type assignment to improve wirelength and routability.
- We propose a new analysis method based on graph theory for understanding the new cell type assignment problem, which is valuable for both physical design and DTCO.

Experimental results demonstrate our superiority over SOTA work



(a) FS. (b) BS w/o timing benefits. (c) BS w/ timing benefits.

Fig. 2 Assumptions about the metal pitches (widths) of FS and BS layers. (a) has a typical layer number and pitches as [6], [14]. (b) is BS based on identical assumption of FS as [14]–[16]. (c) is BS based on Intel 18A setting [6]. DS pins are mostly set on FM0/FM1/BM0/BM1 (orange).

[15]. We reduce wirelength by 21.3%, WNS by 29.2%, TNS by 23.1%, power by 0.6%, design rule violations (DRVs) by 4.358×, and achieve 5.216× speedup. Besides, we successfully routed designs under the settings of Intel 18A, while the default flow without our CTA would all fail.

We organize this paper as follows. In Section II, we demonstrate the preliminaries of our work. In Section III, we explain the details of our algorithms and analysis. In Section IV, we present the results of our work. In Section V, we conclude our work and discuss further work.

II. Preliminaries

We list the main symbols for our work in TABLE II and introduce other preliminaries in this section.

A. DS Metal Layer Distribution

The current DS physical design flow is still rapidly evolving, which results in different BS metal layer distributions being adopted in the academic works [14]–[16] in Fig. 2(b) and Intel 18A [6] in Fig. 2(c). In Fig. 2(b), the works [14]–[16] assume identical layer distribution on FS and BS, whose setting is concluded as BS w/o timing benefits. The identical layer distribution prevents us from easily gaining timing benefits by solely assigning nets on FS or BS. In Fig. 2(c), Intel 18A [6] set BS metal layers with larger pitch (width) to make nets on BS have overwhelming timing benefits, which is concluded as BS w/ timing benefits. To accommodate the DS pins design scheme, we also set two fine-grained BS layers, BM0/BM1, to implement cell pins for Intel 18A setting as Fig. 2(c).

B. Intra-Cell DS Pin Configurations

We define a library cell as **Conflict-type** and **No-conflict-type** according to the types equipped for them. A type in a library cell is uniquely defined as a combination of the sides of input pins, where each input pin is FS or BS. For a library cell with p input pins, there are exactly 2^p possible types in total. We define a library cell with p input pins as **No-conflict-type** when candidate types equal to 2^p. We define a library cell with p input pins as **Conflict-type** when the types, all input pins being on the same side, are missing. An example of these definitions is shown in Fig. 3. The **No-conflict-type** design scheme is preferred when a large area budget is provided for cell design [8], since pins should consume area for accessibility. However, for area-optimized library designs [8], pins have to be assigned on different sides to meet the target area of cells [8], which is **Conflict-type** design scheme.

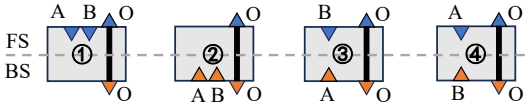


Fig. 3 Cross-section view of the intra-cell 3D positions of flexible input pins and duplicated output pin. A library cell is in **Conflict-type** when it contains ③ and ④; and a library cell is in **No-conflict-type** when it contains ①, ②, ③, and ④.

TABLE II Main symbols in this work.

Symbol†	Description
E	Set of nets in the design.
E_f	Set of FS nets and the FS part of DS nets in the design.
V	Set of cells in the design.
L_v	Set of library cells belonging to cell $v \in V$.
F	FS layers: $\{f_0, f_1, \ldots, f_{m-1}\}$.
B	BS layers: $\{b_0, b_1, \ldots, b_{n-1}\}$.
$r_g(e)$	Route demand mapped to grid $g \in$ GCells of net $e \in E$.
$\mathrm{D}_g(E_f)$	Total FS route demand mapped to grid $g \in$ GCells of nets E_f.
C_f	FS route capacity.
$\mathrm{WL}(E)$	Wirelength of virtual nets or single-sided part of DS nets E.
$\mathrm{DWL}(E)$	Wirelength of real DS nets E.
$x_{v,l}$	Selection of $l \in L_v$ for cell $v \in V$.
$y_{v,e,l}$	Side of pin (v,e) in library cell $l \in L_v$.
$y_{v,e}$	Side of pin (v,e): $y_{v,e} = \sum_{l \in L_v} y_{v,e,l} \cdot x_{v,l}$.
z_e	Virtual net variable indicating preferred side for net e.

† E_b, $\mathrm{D}_g(E_b)$, and C_b: defined by updating FS→BS in E_f, $\mathrm{D}_g(E_f)$, and C_f.

C. Route Demand Model

We employ the rectangular uniform wire density (RUDY) model [17] to estimate the route demand on the layout. By RUDY model, each net e has a constant congestion value $r(e)$ when pin positions are determined. The value $r(e)$ is mapped to grid cells (GCells) using $r_g(e)$, which is defined by the width w_e and height h_e of the bounding box as follows:

$$r_g(e) \propto \mu_e \cdot \frac{w_e + h_e}{w_e \cdot h_e}. \tag{1}$$

In (1), μ_e represents the wire density based on average wire width. By replacing $w_e + h_e$ in the numerator with w_e or h_e, the horizontal and vertical RUDY can be defined. Furthermore, the total route demand of each grid $g \in$ GCells is $\mathrm{D}_g(E) = \sum_{e \in E} r_g(e)$.

D. Problem Formulation

Problem 1 (Cell Type Assignment)**.** *Given a placed netlist, cell library with DS pins, and FS/BS metal layer settings, find the optimal cell types to minimize wirelength and improve routability.*

III. Algorithms

A. Overview

Our algorithm comprises two-level optimization stages, including virtual net planning (VNP) and virtual-net-guided CTA, which are illustrated in Fig. 4. By solving VNP, we achieve the preferred positions for nets on FS or BS, considering net overlap and layer settings. Guided by the VNP result, we assign cell types by combining preferred net sides and DS wirelength together for a comprehensive decision. Considering the number of variables in Eq. (2), the efficiency of the algorithm is important; therefore, integer linear programming (ILP) is not suitable for this problem.

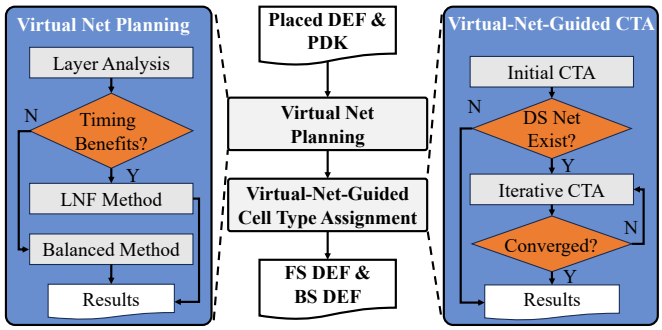
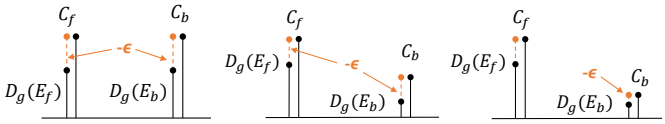


Fig. 4 Overview of our algorithm framework.



(a) w/o timing benefits, $C_f = C_b$. (b) w/o timing benefits, $C_f > C_b$. (c) w/ timing benefits, $C_f > C_b$.

Fig. 5 Route demand and capacity modeling under different configurations of BS metal layers. (a) and (b) represent academic settings. (c) represents Intel 18A settings.

Before the algorithm details, we introduce a new general mathematical formulation for CTA as follows:

$$\min_{X} \quad \alpha_f \mathrm{WL}(E_f; X) + \alpha_b \mathrm{WL}(E_b; X) + \epsilon, \tag{2a}$$

$$s.t. \quad \sum_{l \in L_v} x_{v,l} = 1, \ x_{v,l} \in \{0,1\}, \ \forall\, v \in \mathrm{V}, \ l \in L_v, \tag{2b}$$

$$\mathrm{D}_g(E_f; X) - C_f \le \epsilon, \ \forall g \in \mathrm{GCells}, \tag{2c}$$

$$\mathrm{D}_g(E_b; X) - C_b \le \epsilon, \ \forall g \in \mathrm{GCells}. \tag{2d}$$

α_f and α_b represent timing factors induced by different FS and BS metal layer distributions. ϵ represents the overflow bound of route demand. The route capacities, C_f and C_b, are modeled by projecting the route tracks onto the F and B metal layers to FS and BS GCells with length information, thereby making route capacity comparable to route demand.

Besides, we introduce the properties of the two BS layers in Fig. 2 as follows: (1) BS w/o timing benefits: The FS and BS distributions of metal-layer widths are almost identical. When the ratio of FS/BS metal layers is adjusted, e.g., $C_f = C_b$ and $C_f > C_b$, the goal is still to balance the overflow on FS and BS. (2) BS w/ timing benefits: The BS metal layers have explicit timing benefits over FS metal layers, but the BS route capacity can be much smaller, i.e., $C_f > C_b$. The algorithm should fully leverage BS metal layers to gain timing benefits and optimize BS overflow to maintain routability.

B. Virtual Net Planning

As the original problem of CTA is difficult to solve due to the binary constraint on $x_{v,l}$, we introduce the virtual net variable z_e that indicates the preferred side for each net e as Fig. 6. VNP only takes $\{z_e\}$ as variables, which means all nets can be assigned to FS or BS independently and may be virtual compared with a real DS net. We also formulate the VNP problem as follows:

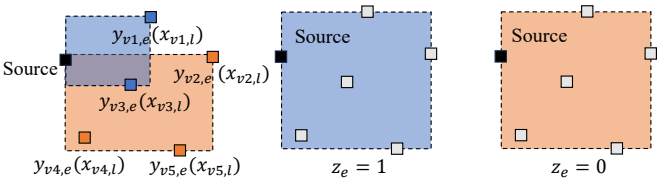
$$\min_{Z} \quad \alpha_f \mathrm{WL}(E_f; Z) + \alpha_b \mathrm{WL}(E_b; Z) + \epsilon, \tag{3a}$$

$$s.t. \quad z_e \in \{0,1\}, \quad \forall e \in E, \tag{3b}$$

$$\mathrm{D}_g(E_f; Z) - C_f \le \epsilon, \ \forall g \in \mathrm{GCells}, \tag{3c}$$

$$\mathrm{D}_g(E_b; Z) - C_b \le \epsilon, \ \forall g \in \mathrm{GCells}. \tag{3d}$$

The VNP in Eq. (3) is constructed by a relaxation of CTA in Eq. (2).



(a) Real DS Net. (b) Virtual Net.

Fig. 6 Real DS net in (a) is determined by multiple variables, while the virtual net in (b) is determined by one variable.

This relationship is further explained in Section III-D.

1) Algorithm Details: We design an efficient algorithm for VNP, considering BS w/o timing benefits and BS w/ timing benefits, as shown in Fig. 5. At the beginning of Alg. 1, we first initialize three route demand maps and sort all GCells to determine their priority (Line 1∼2). To scale C_b, we define

$$htb = min\_width(b_{2\sim n-1}) > ave\_width(f_{2\sim m-1}). \tag{4}$$

We also set assignment criteria $assignBS$ as follows:

$$assignBS = \begin{cases} \mathrm{D}_g(E_b) < C_b & , htb = true, \\ \mathrm{D}_g(E_b) < \mathrm{D}_g(E_f) - C_f + C_b & , \text{otherwise}. \end{cases} \tag{5}$$

Large-net-first (LNF) method: This method is used when BS has timing benefits, e.g., Intel 18A. This approach prioritizes assigning large nets (higher HPWL) to BS layers to maximize timing benefits. Since the RUDY model underestimates routing demand for large nets, we scale the BS layer capacity C_b by a factor η (Line 3) to mitigate underestimation risks, thus improving the timing and routability of BS simultaneously.

Balanced method: This method handles BS w/o timing benefits, where the weighted wirelength remains unchanged during planning. The focus shifts to minimizing overflow ϵ by balancing route demand between FS and BS. The term $C_f - C_b$ quantifies the capacity difference between FS and BS. Unlike the LNF method, this approach mitigates underestimation risks by dynamically distributing large nets across both sides.

2) Complexity Analysis: Considering the grid route demand map size is $|\mathcal{D}|$, the total number of nets is N, the complexity of Alg. 1 is $O(|\mathcal{D}|log(|\mathcal{D}|) \times N)$. Conventionally, $|\mathcal{D}| \ll N$ are set to make our algorithm efficient for large design cases.

C. Virtual-Net-Guided Cell Type Assignment

In Alg. 2, we assign cell types by combining preferred net sides and DS wirelength together for a comprehensive decision in initial CTA (Init-CTA) and iterative CTA (Iter-CTA).

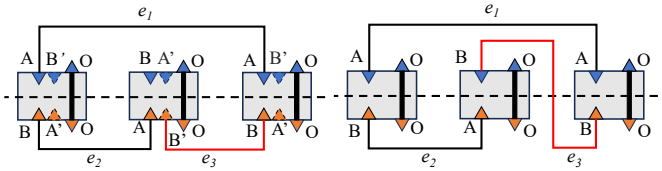
1) Initial Cell Type Assignment: The Init-CTA (Line 1∼3 in Alg. 2) aims to find a solution that takes advantage of VNP results and the wirelength of all virtual nets to decide the cell assignment results as follows:

$$l_v^* = \arg\min_{l \in L_v} \sum_{e \in E_v} \mathrm{WL}(e) \cdot |y_{v,e,l} - z_e|. \tag{6}$$

Algorithm 1: Virtual Net Planning.

Input: Route demand map: FS $\mathcal{D}_f = [\mathrm{D}_g(E_f)]^{|\mathrm{Gcells}|}$, BS $\mathcal{D}_b = [\mathrm{D}_g(E_b)]^{|\mathrm{Gcells}|}$, unassigned $\mathcal{D}_u = [\mathrm{D}_g(E)]^{|\mathrm{Gcells}|}$, and nets RUDY $\{r_g(e)\}$.
Output: Net planning results $Z = \{z_e | e \in E\}$.

1 Initialize $\mathcal{D}_f \leftarrow 0$, $\mathcal{D}_b \leftarrow 0$, and $\mathcal{D}_u \leftarrow \sum_{e \in E} r_g(e)$.
2 Sort $\mathcal{D}_u$ by $\mathrm{D}_g(E)$ in descending order as $Sorted(Gcells)$.
3 Scale $C_b \leftarrow C_b \times \eta$ if *hbt=true*.
4 Set *assignBS* by Eq. (5).
5 **for** g *in* $Sorted(Gcells)$ **do**
6 Update $\mathrm{D}_g(E_f)$ and $\mathrm{D}_g(E_b)$ by assigned nets on g.
7 Sort unassigned nets by $\mathrm{WL}(e)$ in descending order.
8 **for** $e \in \{e \mid \mathrm{isUnAssigned}(e)$ & e *intersect* $g\}$ **do**
9 **if** *meets assignBS* **then**
10 $\mathrm{D}_g(E_b) \leftarrow \mathrm{D}_g(E_b) + r_g(e)$, $z_e \leftarrow 0$.
11 **else**
12 $\mathrm{D}_g(E_f) \leftarrow \mathrm{D}_g(E_f) + r_g(e)$, $z_e \leftarrow 1$.
13 Update e as assigned.



(a) Virtual Net Planning. (b) Initial Cell Type Assignment.

Fig. 7 (a) Virtual net planning of net e_1, e_2, and e_3, and (b) initial cell type assignment with cells in **Conflict-type**. (A', B') and (A, B) in (a) refer to two candidate types.

The goal of the optimization problem in Eq. (6) is to search cell type $l \in L_v$ for each cell v to align with z_e of the connected virtual net. The lower bound of the objective in Eq. (6) equals zero explicitly, which can be achieved if every pin side $y_{v,e,l}$ equals the virtual net result z_e. Even if the lower bound of the objective can not be reached, Eq. (6) still makes the virtual net with a larger wirelength more likely to be aligned with the connected pin.

In Fig. 7, we show that Init-CTA determines the cell types by the VNP results. We find the pins in the net e_1 with the largest wirelength are preserved on the same side to optimize the total wirelength. By combining VNP in Eq. (3) and Init-CTA in Eq. (6), our method efficiently finds a relatively good solution without enforcing any cell type.

2) Iterative Cell Type Assignment: The Iter-CTA aims to optimize cell types based on real DS wirelength and pin dependencies. At the beginning of each iteration, we update the pin sides and compute the DS wirelength for all nets (Line 5∼6). Then we update cell types if doing so reduces the DS wirelength.

Pin increment computation refers to the DS wirelength increase when flipping one pin to the other side, as follows:

$$\delta_{v,e}^{(k)} = \mathrm{DWL}(e)|_{\text{if } y_{v,e}^{(k)} \leftarrow \neg y_{v,e}^{(k)}} - \mathrm{DWL}(e). \tag{7}$$

As Eq. (7), if $\delta_{v,e}^{(k)}$ is less than zero, the flipping operation has the potential to decrease DS wirelength. We record all pin increment values for each cell v. Besides, we use additional conditions to

Algorithm 2: Virtual-Net-Guided Cell Type Assignment.

Input: Cell libraries L, cells V, nets E, VNP results Z.
Output: Cell assignment results $\{\mathrm{x}_{v,l}\}$.

1 **for** $v \in \mathrm{V}$ **do**
2 Get l_v^* by Eq. (6).
3 $x_{v,l_v^*} \leftarrow 1$ and other $x_{v,l} \leftarrow 0$.
4 **while** *total DS wirelength decrease ratio* $> 1 \times 10^{-3}$ **do**
5 Update pin sides $\{y_{v,e}\}$ by definition in TABLE II.
6 Compute net DS wirelength $\{\mathrm{DWL}(e)\}$.
7 Compute pin increment $\{\hat{\delta}_{v,e}^{(k)}\}$ by Eq. (7) and Eq. (8).
8 **for** $v \in \mathrm{V}$ **do**
9 Get $l_v^{*(k)}$ by Eq. (9).
10 **if** *optimal value of Eq.* (9) < 0 **then**
11 $x_{v,l_v^{*(k)}} \leftarrow 1$ and other $x_{v,l} \leftarrow 0$.

modify $\delta_{v,e}^{(k)}$ to $\hat{\delta}_{v,e}^{(k)}$ as follows:

$$\hat{\delta}_{v,e}^{(k)} = \begin{cases} 0, & \delta_{v,e}^{(k)} < 0 \ \& \ y_{v,e}^{(k)} = z_e, \\ \delta_{v,e}^{(k)}, & \text{otherwise.} \end{cases} \tag{8}$$

The condition, $\delta_{v,e}^{(k)} < 0$ & $y_{v,e}^{(k)} = z_e$, means if the pin flipping decreases DS wirelength, $\delta_{v,e}^{(k)}$ will be reset to zero for preventing this pin from flipping to the opposite of z_e as much as possible.

Cell type update decides cell types based on different pin increments, which is defined as an optimization problem for each cell v as follows:

$$l_v^{*(k)} = \arg\min_{l \in L_v} \sum_{e \in E_v} \hat{\delta}_{v,e}^{(k)} \cdot |y_{v,e,l} - y_{v,e}^{(k)}|. \tag{9}$$

In the objective of Eq. (9), $y_{v,e}^{(k)}$ is the pin side in the k-th iteration, which is used to compare with $y_{v,e,l}$. If $\hat{\delta}_{v,e}^{(k)} > 0$, keeping $y_{v,e,l}$ equals to $y_{v,e}^{(k)}$ is preferred. If $\hat{\delta}_{v,e}^{(k)} < 0$, flipping $y_{v,e}^{(k)}$ to the opposite side is preferred. Therefore, Eq. (9) can hold the good pin assignment while flipping the other pin assignments. Besides, we also check the optimal value of Eq. (9) (Line 10) for cell type update, which prevents algorithms from unnecessary updating. We set the stop criterion by the total DS wirelength decrease ratio less than 1×10^{-3} (Line 4), when the decrease of DS wirelength is not significant.

3) Complexity Analysis: Let total iterations as I, maximum pins of net e as P_e, the complexity of CTA is $O(|\mathrm{V}|IP_e^2)$. P_e^2 results from pin increment computation. Even for large designs, the average pins in a net is small, which makes our algorithm efficient.

D. DTCO Insights from Conflict Graph

In this part, we analyze the relationship between VNP and CTA using a conflict graph and further provide DTCO insights.

1) Relationship between VNP and CTA: Here, we only discuss two-input-pin cells in **Conflict-type** or **No-conflict-type** for convenience. The analysis of multi-input-pin (> 2) cells can also be investigated by pairwise relationships among pins, similarly. We define Node e_i and Edge (e_i, e_j) in the conflict graph as follows:

Definition 1. *Node* $e_i \triangleq$ *virtual net variable* z_{e_i}.

Definition 2. *Edge* $(e_i, e_j) \triangleq$ *cell, driven by* e_i *and* e_j*, is* ***Conflict-type***.

For the conflict graph, we can always achieve a two-coloring scheme [18], which could be legal or illegal. A legal edge in a two-

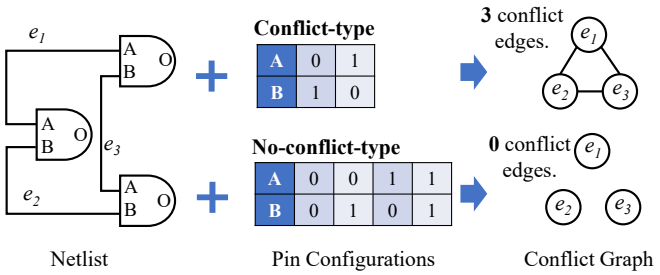


Fig. 8 Conflict graph is built by a netlist and associated cell library. Cell types, in **Conflict-type** or **No-conflict-type**, decide edge number in conflict graphs.

coloring scheme means the associated cell has a legal assignment by (z_{e_i}, z_{e_j}). Therefore, a legal two-coloring scheme is equivalent to that all cells have legal assignments by applying (z_{e_i}, z_{e_j}), the goal as Init-CTA, which is a solution to CTA without DS nets. As described in Section III-B, VNP admits virtual net variables z_e assigned to 1 or 0 independently. Therefore, a solution to VNP is equivalent to a two-coloring of the conflict graph. That is, if the conflict graph is two-colorable, a solution to VNP is equivalent to a solution to CTA without DS nets.

Meanwhile, the number of two-coloring schemes on the conflict graph restricts the solution space of VNP. Since the nodes and edges in the conflict graph are completely defined by the netlist and cell library, the gap between VNP and CTA is an inner property that can be analyzed by the two-coloring problem on the conflict graph. By fully relaxing the solution space of the two-coloring problem, i.e., ensuring all cells in **No-conflict-type**, no edges exist in the conflict graph. Since VNP requires an independent assignment of z_e, it can be viewed as constructed based on relaxation.

Besides, a conflict graph without a legal two-coloring scheme implies the existence of odd cycles [18], i.e., cycles with odd nodes. Applying our conflict graph construction method and odd-cycle detection could yield meaningful insights into DS nets, which are unique and critical in the emerging DS physical design flow.

2) DTCO Insights: From the analysis by conflict graph, we observe that pin configurations are critical for the existence of DS nets, which could degrade wirelength. Meanwhile, considering the heavier effort in design cells, supporting all pin configurations for all cells could be very difficult. We provide two insights for DTCO as follows: (1) The evaluation of the cell library could adopt the number of odd cycles in the synthesized netlist as a new metric to decide which pin configurations are better. (2) Physical design could resynthesize some critical parts in the netlist or perform odd-cycle-aware buffer insertion to reduce odd cycles.

IV. Experimental Results

We implement our algorithm framework in C++ as `Ours` in TABLE III. In the algorithm, we use HPWL as the wirelength model and RUDY as the wirelength route demand model for optimization, which can be replaced with more sophisticated modeling methods. We perform the experiments on the Linux platform with a 192-core 2.40GHz Intel(R) Xeon(R) Platinum 8260 and 512GB memory. We use Design Compiler for logic synthesis in 1 GHz and Innovus for other stages following the flow in Fig. 1. We use StarRC for parasitic extraction and PrimeTime for timing and power evaluation. We prepare PDK in the same way as [14]. We obtain designs from OpenROAD [19] and ChipBench [20], including `mor1kx(C1)` (nets: 135.9K, cells: 135.8K), `ariane133(C2)` (nets: 205.0K, cells: 204.3K), `FPGA-CAN(C3)` (nets: 184.6K, cells: 184.6K), and `ariane136(C4)` (nets: 210.7K, cells 208.6K).

A. Comparison with the SOTA Method

We perform the experiments on the benchmarks and compare the results with the SOTA method [15]. We set six FS metal layers ($f_{1\sim6}$) and three BS metal layers ($b_{1\sim3}$) for routing, which is more realistic than the assumption of identical FS and BS metal layers. We set placement utilization as 80%. In TABLE III, we use **No-conflict-type** for comparison because this configuration provides all types for algorithms, which makes the comparison fair. Compared with [15], `Ours` optimizes RWL by 1.213×, #DRV by 4.358×, PWR by 1.006×, WNS by 1.292×, and TNS by 1.231×. Besides, our algorithm achieves 5.216× speedup over [15]. We observe that larger designs benefit more from `Ours` in WNS and TNS, as the congestion in larger designs is more severe and can be mitigated by shorter wirelength.

We also modify [15] to adapt **Conflict-type**, where we flip pins as many as possible to the opposite side when congestion exists on one side of this cell. The results in TABLE III demonstrate that our algorithm optimizes RWL by 1.170×, WNS by 1.203×, TNS by 1.204×, and #DRV by 74.200× on average. Besides, our algorithm achieves 4.528× speedup over [15].

B. DTCO Study with Flexible Metal Layers

We now conduct a further experiment to explore DTCO opportunities using our CTA algorithm and DS flow. The target of this experiment is to compare the impacts of layer configurations on the quality of results (QoR) of DS physical designs. We implement clock routing on BS and set $\eta = 0.1$ because large nets are preferred for BS. We set FS metal layers with pitches of $f_{1\sim6}$: 32/26/35/35/42/76 nm. We run results under two DS settings: one following the Intel 18A layer configuration (pitches of $b_{1\sim4}$: 48/104/300/480 nm) and the other following a scaled BS metal layer configuration (pitches of $b_{1\sim4}$: 32/76/76/126 nm). We set the placement utilization as 70% and all cells in **No-conflict-type**.

We compare `Ours` with `FS`, which is a method that implements all signal nets on FS, to show the timing benefits. We cannot compare `Ours` with [15] because it cannot be routed within 24 hours due to too many DRVs on the BS. The reason is that [15] fails to model the extremely imbalanced route capacity on FS and BS in the Intel 18A setting and incurs unacceptable route demand on BS. The BS wirelength of `C4` by `Ours` only takes up 3.3%, while [15] has over 10× more than `Ours`. Compared with `FS`, `Ours` improves WNS by 1.219× and TNS by 1.162× and incurs more DRVs. The increased DRVs indicate that integrating DS pins [14] with the Intel 18A setting could degrade routability, requiring more DTCO attempts. In the scaled BS metal layer configuration, we can achieve further timing improvements by reducing the metal layer pitches. We infer that the bad routability in BS also prevents nets from gaining timing benefits, even though BS pitches are larger. This type of DTCO exploration is significant for the pathfinding of DS physical design, which can only be achieved when the CTA tool can be applied in flexible settings.

C. Runtime Breakdown

We show the runtime breakdown of `Ours` on the design `C2` for cells in **No-conflict-type** and **Conflict-type**. **No-conflict-type** has less runtime than **Conflict-type** due to the elimination of DS nets by combining our VNP and Init-CTA methods.

TABLE III Comparison with the SOTA method [15]. FS layers: $f_{1\sim6}$ and BS layers: $b_{1\sim3}$.†

Design	[15] RWL↓ (μm)	WNS↑ (ns)	TNS↑ (ns)	PWR↓ (mW)	#DRV↓	RT↓ (s)	Ours RWL↓ (μm)	WNS↑ (ns)	TNS↑ (ns)	PWR↓ (mW)	#DRV↓	RT↓ (s)
	Based on cells in **No-conflict-type**.											
C1	200170	-11.530	-76026.480	31.300	**2**	43.573	**171093**	**-8.240**	**-59046.400**	**29.200**	14	**18.530**
C2	358542	-7.540	-50578.270	**2037.400**	173	147.146	**287643**	**-5.430**	**-39787.510**	2057.700	**21**	**19.526**
C3	239794	-13.730	-248519.750	110.800	**4**	112.790	**205965**	**-11.430**	**-206171.470**	**97.100**	8	**33.101**
C4	370855	-7.340	-52898.620	2078.700	52	170.646	**299655**	**-5.970**	**-42785.500**	**2050.800**	**10**	**19.754**
Ratio	1.213	1.292	1.231	1.006	4.358	5.216	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Based on cells in **Conflict-type**.											
C1	202234	-12.480	-77074.710	31.800	32	44.108	**172192**	**-8.690**	**-59260.520**	**29.300**	**0**	**24.928**
C2	368866	-7.730	-53967.850	**2034.300**	367	264.811	**310366**	**-6.320**	**-41368.540**	2052.100	**4**	**28.060**
C3	237116	-13.220	-250180.050	111.100	6	112.043	**210133**	**-12.500**	**-211414.770**	**98.600**	**4**	**42.837**
C4	380892	-6.860	-48351.150	2075.500	337	141.349	**323904**	**-5.970**	**-44772.000**	**2073.200**	**2**	**28.353**
Ratio	1.170	1.203	1.204	1.000	74.200	4.528	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**

† Routed wirelength (RWL) and the number of design rule violations (#DRV) are reported by Innovus. Worst Negative Slack (WNS), Total Negative Slack (TNS), and power (PWR) are reported by PrimeTime. ↓ means "lower is better". ↑ means "higher is better".

TABLE IV DTCO results based on different BS metal layers.

Design	#DRV FS	#DRV Ours	WNS (ns) FS	WNS (ns) Ours	TNS (ns) FS	TNS (ns) Ours
Intel 18A BS metal layer configuration.						
C1	**2**	10	-7.72	**-7.06**	-47909.57	**-46172.99**
C2	84	**53**	-7.35	**-6.89**	-46414.81	**-44081.09**
C3	**0**	32	-19.7	**-13.78**	-241916.22	**-194485.86**
C4	**8**	15	-5.44	**-5.25**	-43045.17	**-41717.48**
Ratio	0.855	1.000	1.219	1.000	1.162	1.000
Scaled BS metal layer configuration.						
C1	1	**0**	-6.07	**-5.45**	-44064.42	**-39825.01**
C2	**3**	4	-5.46	**-5.19**	-41722.46	**-38982.29**
C3	**0**	5	-15.51	**-11.06**	-240371.04	**-138633.24**
C4	**0**	2	**-5.56**	-5.72	-33877.29	**-31794.56**
Ratio	0.364	1.000	1.189	1.000	1.445	1.000

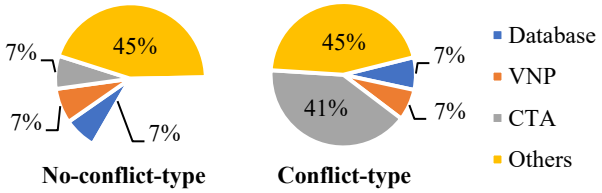


Fig. 9 Runtime breakdown of Ours on the design C2.

V. Conclusion

In this work, we formulate a new mathematical problem for double-sided pin redistribution and propose a virtual-net-guided cell type assignment to solve it. We also propose a graph-theoretic analysis method for DS designs and DTCO. Based on evaluations using commercial tools, our algorithms improve the wirelength by 21.3%, the overall timing by 23.1%, and the power by 0.6%, and reduce the DRVs by 4.358× on average compared with the SOTA method. Meanwhile, we further successfully perform DTCO exploration based on Intel 18A settings using our tools, while the SOTA method fails. We believe our work can spur more research into double-sided physical design. In the future, we will investigate timing-driven methods to improve performance.

References

[1] J. Ryckaert *et al.*, "The complementary fet (cfet) for cmos scaling beyond n3," in *2018 IEEE Symposium on Vlsi Technology*. IEEE, 2018, pp. 141–142.

[2] S. Subramanian *et al.*, "First monolithic integration of 3d complementary fet (cfet) on 300mm wafers," in *2020 Ieee Symposium on Vlsi Technology*. IEEE, 2020, pp. 1–2.

[3] P. Schuddinck *et al.*, "Ppac of sheet-based cfet configurations for 4 track design with 16nm metal pitch," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2022, pp. 365–366.

[4] O. Zografos *et al.*, "Design enablement of cfet devices for sub-2nm cmos nodes," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 29–33.

[5] C.-K. Cheng *et al.*, "A routability-driven complimentary-fet (cfet) standard cell synthesis framework using smt," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–8.

[6] K. e. a. Fischer, "Intel 18a platform technology featuring ribbonfet (gaa) and powervia for advanced high-performance computing," in *2025 Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2025, pp. 1–3.

[7] "Tsmc a16 technology."

[8] H. Lu *et al.*, "First experimental demonstration of self-aligned flip fet (ffet): A breakthrough stacked transistor technology with 2.5 t design, dual-side active and interconnects," in *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2024, pp. 1–2.

[9] T. Kim, "Physical design challenges for design technology co-optimization," in *Proceedings of the 2025 International Symposium on Physical Design*, 2025, pp. 20–21.

[10] N. E. Bethur *et al.*, "Gnn-assisted back-side clock routing methodology for advance technologies," *Proceedings of the 61st Design Automation Conference*, 2024.

[11] X. Jiang *et al.*, "A systematic approach for multi-objective double-side clock tree synthesis," in *IEEE/ACM DAC*, 2025.

[12] T.-C. Lin *et al.*, "An effective netlist planning approach for double-sided signal routing," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2024, pp. 288–293.

[13] S. Liao *et al.*, "First demonstration of monolithic cfet inverter at 48nm gate pitch toward future logic technology scaling," in *2024 IEEE International Electron Devices Meeting (IEDM)*, 2024, pp. 1–4.

[14] H. Lu *et al.*, "A tale of two sides of wafer: Physical implementation and block-level ppa on flip fet with dual-sided signals," *2025 Design, Automation & Test in Europe Conference & Exhibition*, 2025.

[15] J. Ahn *et al.*, "Design and technology co-optimization utilizing flip-fet (ffet) standard cells," in *IEEE/ACM DAC*, 2025.

[16] S. Choi *et al.*, "Omni 3d: Beol-compatible 3-d logic with omnipresent power, signal, and clock," *IEEE Transactions on Electron Devices*, 2025.

[17] P. Spindler *et al.*, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6.

[18] T. H. Cormen *et al.*, *Introduction to algorithms*, 2022.

[19] T. Ajayi *et al.*, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," *Proc. GOMACTECH*, pp. 1105–1110, 2019.

[20] Z. Wang *et al.*, "Benchmarking end-to-end performance of ai-based chip placement algorithms," *arXiv preprint arXiv:2407.15026*, 2024.