A P&R Co-Optimization Engine for Reducing Congestion

Dongliang Xia
Wenxin Yu*
yuwenxin@swust.edu.cn
Southwest University of Science and Technology
Mianyang, Sichuan, China

Zhaoqi Fu
Zejun Gan
Southwest University of Science and Technology
Mianyang, Sichuan, China

Chengjin Li
Yupeng Zhang
Southwest University of Science and Technology
Mianyang, Sichuan, China

ABSTRACT

Placement and routing (P&R) are two crucial stages in the physical design process to optimize different objectives. For instance, placement often focuses on optimizing the half-perimeter wirelength (HPWL) and estimated congestion while routing attempts to minimize the wirelength and the number of overflows. The misalignment of objectives inevitably leads to a significant decline in solution quality. Therefore, this paper is an efficient P&R co-optimization engine that bridges the gap between placement and routing disparities. To progressively alleviate routing congestion issues, we perform cell movements on cells causing overflows and then re-run the routing process. Comparing our experimental results with CUGR [5], our method reduced 17.1% in overflow, with slight decreases in wirelength and vias.

CCS CONCEPTS

• **Hardware → Electronic design automation**; **Physical design (EDA)**; **Wire routing**; **Placement**.

KEYWORDS

global routing, placement, cell movement

ACM Reference Format:
Dongliang Xia, Wenxin Yu, Zhaoqi Fu, Zejun Gan, Chengjin Li, and Yupeng Zhang. 2024. A P&R Co-Optimization Engine for Reducing Congestion. In *Great Lakes Symposium on VLSI 2024 (GLSVLSI '24), June 12–14, 2024, Clearwater, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3649476.3658796

1 INTRODUCTION

Dividing placement and routing reduces complexity but can lower solution quality due to misaligned objectives and conservative margins. Optimizing placement before routing harms quality while repositioning cells based on routing improves results.

Recently, CUGR [5] introduced a probabilistic cost scheme and utilized multi-level 3D maze routing for network routing. Their experimental results demonstrate the superior performance of their router in the ICCAD'19 Global Routing Contest [6] compared to all participants. To enhance the correlation between placement and routing, existing placement tools often employ different strategies to optimize routing. For example, RePlAce [1] developed an effective super-linear cell inflation technique to alleviate global routing congestion during global placement. NTUplace4dr [4] proposed a novel dynamic cell inflation and congestion elimination model to optimize profitability. These methods rely on a limited number of routing congestion estimates to guide placement rather than producing routing solutions directly. Therefore, the gap between placement and routing remains challenging to bridge directly.

To further promote research in the direction of bridging the gap between placement and routing, ICCAD organized two contests in 2020 and 2021 [2, 3], respectively, focusing on routing with cell movement. Both contests aimed at reducing wirelength on a benchmark without any specific constraints. Peng Zou et al.

However, all these routers apply cell movement, which can further optimize routing resources. Therefore, this paper proposes improving routing by addressing cell movement causing congestion. However, the global routing cases in the ICCAD 2020 competition [2] did not involve overflow. Hence, this paper aims to optimize overflow in cases with overflow, building upon CUGR [5] and utilizing cell movement.

Our contributions mainly include the following key points:

- We propose a P&R co-optimization engine that can effectively reduce overflow in the initial place-and-route (P&R) solution during the global routing stage
- We categorize overflow into two scenarios to locate cells that can contribute to overflow reduction precisely. For each cell, we conduct a gain evaluation through swap movements.
- The experimental results on the ICCAD 2019 competition[6] benchmarks show that our proposed P&R co-optimization engine outperforms the globally advanced router CUGR[5], reducing overflow by 17.1%. Additionally, there is a 0.5% reduction in wirelength and a 2.6% decrease in the number of vias compared to CUGR[5].

The rest of this paper is organized as follows. Section 2 describes the fundamental principles and terminology of the global routing problem. Section 3 introduces the methods proposed. The experimental results are provided in Section 4. Finally, We conclude in Section 5.

2 PRELIMINARIES

2.1 Terminologies

Our algorithm is designed based on the primary concept of overflow; hence, the following section briefly explains the concept of overflow.

*Corresponding Author.

GLSVLSI '24, June 12–14, 2024, Clearwater, FL, USA

ACM ISBN 979-8-4007-0605-9/24/06
https://doi.org/10.1145/3649476.3658796

Capacity : The capacity of an edge $e(u, v)$ represented by $c(u, v)$ is the number of tracks that can pass through that edge, roughly indicating the maximum number of networks that can utilize that edge. the capacity of an edge $V(u, u')$ represented by $c(u, u')$ is the number of tracks that can pass through that via,

Demand : The demand of an edge, $d(u, v)$, is the portion of capacity already used by the routed network. It consists of wire demand and via demand. Wire demand is simply the number of wires passing through the edge, while via demand considers the estimated number of tracks affected by vias in the edge region. The capacity of an edge $V(u, u')$ represented by $d(u, u')$ refers to the number of tracks that are required to pass through that via.

$$d(u, v) = wire(u, v) + 1.5 \times \sqrt{\frac{via(u) + via(v)}{2}} \tag{1}$$

$Overflow$: Equations (2a) and (2 b) depict the overflow, capacity, and demand relationship for an edge or a via.

$$o(u, v) = \begin{cases} d(u, v) - c(u, v) & d(u, v) > c(u, v), \\ 0 & d(u, v) \leq c(u, v). \end{cases} \tag{2a}$$

$$o(u, u') = \begin{cases} d(u, u') - c(u, u') & d(u, u') > c(u, u'), \\ 0 & d(u, u') \leq c(u, u'). \end{cases} \tag{2b}$$

3 PROPOSED ALGORITHMS

Figure 1 illustrates an example of routing based on the existing placement and routing, incorporating cell movement. Following the relocation of cells and the network rerouting, a better P&R solution is achieved with fewer overflow. The process proposed in Figure 2 is depicted, where, in the preprocessing phase, we filter networks with overflows. Based on networks with overflows, we then specifically identify the cells that need to be moved. We finalize the overflow cell movement stage process according to the Overflow-Driven Cell Movement and Rerouting algorithm.

3.1 Preprocessing Phase

In the preprocessing phase, we carefully performed cell movements after the maze routing in CUGR[5] had been completed. The rationale behind this decision is our intention to leverage the guidance

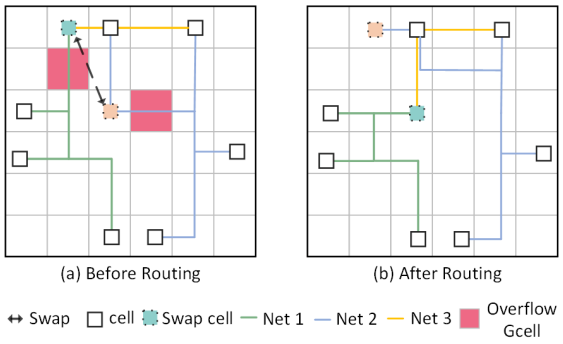

Figure 1: Illustration of the overflow before and after cell movement routing, respectively.In (a) and (b), we clearly illustrate the overflow of (b) less than (a).

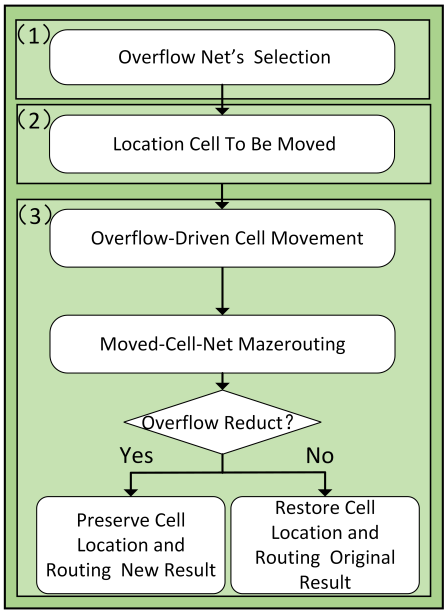

Figure 2: Proposed framework.

information generated by CUGR [5] to optimize the placement of our cells to the fullest extent. By traversing each network, we calculate the capacity of the traversed G-cells based on the generated guidance. The core of this strategy lies in utilizing the guidance information provided by CUGR [5] and precisely identifying networks with overflow by calculating the capacity of each traversed G-cell.

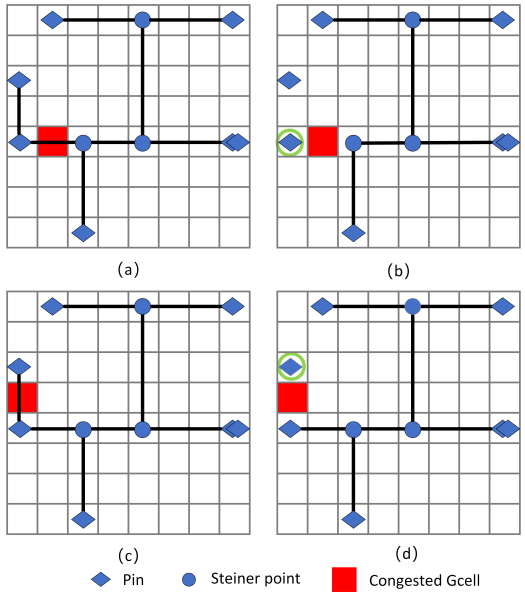

Figure 3: Two overflow cases considered in our flow. The diamond and circle symbols represent pin and Steiner points. (a) and (b) illustrate the first scenario, where congestion occurs between the pins and Steiner points. (c) and (d) depict the second scenario, where congestion occurs between two pins.

3.2 Location Cell To Be Moved

In this section, our focus is directed toward dissecting networks with overflow and analyzing the cells causing the overflow. We delve into two primary scenarios: congestion between a pin and a Steiner point and congestion between two pins.

When congestion occurs between two pins in the same network, we advocate prioritizing cells with more minor degrees. After the relocation of cells with more minor degrees and subsequent rewiring, the number of connections required for the network decreases, thereby reducing the potential risk of congestion. In contrast, moving cells with more significant degrees may lead to more complex routing arrangements, increasing the likelihood of congestion.

In the first scenario, judicious cell movement aims to reduce congestion while maintaining the stability of the connected network. In the second scenario, we are inclined to choose cells with lower degrees for relocation to alleviate the risk of triggering new congestion.

Algorithm 1 gives a set of networks with overflow for each network in this set, a post-order traversal of its topological structure tree is performed. During this traversal, two scenarios may arise: If the current node and its parent node are both pins, a comparison of the degrees of these two nodes is conducted. The node with the more minor degree selected and the corresponding cell is added to the overflow cell set(line 4-8). If the current node or its the parent node is a pin, the pin node is chosen, and the cell associated with the minor degree is added to the overflow cell set(lines 15-19).

Algorithm 1 Location Cell To Be Moved

Input: Set of nets $N_{overflow}$
Output: Set of Cell $C_{overflow}$

1: Create set of Cell $C_{overflow}$
2: **for** each n in $N_{overflow}$ **do**
3: **for** Post-order traversal of $n.topotree$ (routing result before guide) **do**
4: **if** This node n and its parent node p_n both is pin **then**
5: $d_n \leftarrow getdegree(n)$
6: $d_{p_n} \leftarrow getdegree(p_n)$
7: **if** $d_n < d_{p_n}$ **then**
8: $c_{p_n} \leftarrow getcell(p_n)$
9: $C_{overflow}.add(c_{p_n})$
10: **else**
11: $c_n \leftarrow getcell(n)$
12: $C_{overflow}.add(c_n)$
13: **end if**
14: **end if**
15: **if** This node n and its parent node p_n have one pin between them **then**
16: the node n that is a pin
17: $c_n \leftarrow getcell(n)$
18: $C_{overflow}.add(c_n)$
19: **end if**
20: **end for**
21: **end for**
22: **return** $C_{overflow}$

Algorithm 2 Overflow-Driven Cell Movement and Rerouting

Input: Set of Cell $C_{overflow}$, Set of nets N_c associated with Cell $c \in C_{overflow}$, Set of nets N_i associated with Cell $c \in C_{overflow}$
Output: Update Guide G_{N_i} of N_i and Psition of $C_{overflow}$

1: **for** each $c_1(Unmarked)$ in $C_{overflow}$ **do**
2: $c_2 \leftarrow getclosestcell(c, C_{overflow})$
3: **for** each c *in* c_1, c_2 **do**
4: $P_{or} \leftarrow getOriginalPsition(c)$
5: **for** each n *in* N_c **do** //net connected to c
6: $G_{or} \leftarrow getOriginalGuide(n)$
7: **end for**
8: $Overflow_{or} \leftarrow getOverflow(N_c)$
9: **if** $c == c_1$ **then**
10: $P_c = getcellPsition(c_2)$
11: **else**
12: $P_c = getcellPsition(c_1)$
13: **end if**
14: Set of nets N_c of associated with c do 3D MazeRouting
15: $NewOverflow_{change} \leftarrow getoverflow(N_c)$
16: **if** $NewOverflow_{change} \geq Overflow_{or}$ **then**
17: $P_c \leftarrow P_{or}$
18: **for** $n \in N_c$ **do**
19: $G_n \leftarrow G_{or}$
20: **end for**
21: **end if**
22: **end for**
23: Marked c_1 and c_2
24: **end for**

3.3 Overflow-Driven Cell Movement and Rerouting

Faced with challenges that cannot be adequately addressed through conventional wiring strategies, the introduction of local cell mobility allows for localized improvements to the existing placement. Identifying the specific cells that necessitate relocation becomes a pivotal aspect of this strategy. However, the significance lies in the identification process, the meticulous selection of optimal positions for these cells, and the careful consideration of cell layout to ensure effective congestion alleviation.

Our strategy involves moving cells of the same size that are close to each other. When cells of the same size move mutually, there is no need to adjust the positions of surrounding cells. This approach ensures that the network structure of surrounding cells remains unchanged, preserving the relative stability of the network. In contrast, moving cells of different sizes may provide a more extensive solution space. Still, they can result in rerouting the network of the moved cells, disrupting the overflow stability of the original network.

The purpose of choosing cells close is to ensure that the total wirelength of the network does not increase significantly, thereby maintaining network efficiency. After the movement is complete, connections between networks need to be established, and the overflow values of the networks are calculated. We compare the overflow value of the moved network with the original network's overflow value to determine if the network performance has improved.

Table 1: Sorce(Wirelength*0.5+Vias*4+TOF*500),Wirelength,Vias,TOF(edge overflow and vias overflow)and runtime(s) on the ICCAD 2019 Benchmark, Comparison with CUGR [5].

Banchmark	Score		Wirelength		Vias		TOF		Runtime	
	CUGR [5]	Ours	CUGR [5]	Ours	CUGR [5]	Ours	CUGR [5]	Ours	CUGR [5]	Ours
ispd19_test4	23,023,900	22,893,700	29,020,800	28,709,900	650,617	635,375	2,553	2,121	357	394
ispd19_test5	3,862,990	3,771,100	4,776,480	4,771,320	118,126	116,047	438	283	22	36
ispd19_test10	167,655,000	167,479,000	270,243,000	270,115,000	8,054,740	8,050,940	161	110	2,640	3,400
ispd18_test5_metal5	18,755,000	18,434,600	27,907,900	27,483,400	803,000	770,411	539	486	422	797
ispd18_test8_metal5	42,894,300	41,946,600	64,683,600	63,526,200	1,955,250	1,870,500	1,017	775	1491	2,915
ispd18_test10_metal5	44,207,500	42,754,200	70,925,600	69,370,100	2,076,680	1,970,680	183	89	1737	2381
ispd19_test7_metal5	68,001,300	67,619,900	107,010,000	106,892,000	3,071,240	3,040,200	646	555	1520	3016
ispd19_test8_metal5	114,569,000	113,707,000	177,692,000	177,694,000	5,599,040	5,302,280	1,142	1,121	3171	5058
Ratio	1.000	0.991	1.000	0.995	1.000	0.974	1.000	0.829	1.000	1.584

Algorithm 2 starts by selecting an unmoved cell 1 and finding the closest cell 2 of equal size. It backs up guidance information, current positions of cells 1 and 2, and overflow status (line 4-8). Then, it exchanges the positions of cells 1 and 2 to simulate maze routing(line 9-13). After adjustment, a new overflow value is calculated and compared with the initial value. If the new value exceeds the initial one, indicating no improvement, it restores the original positions and guidance. Otherwise, it saves the new positions and guidance to record the optimization results(line 14-21).

4 EXPERIMENTAL RESULTS

Our P&R co-optimization engine, implemented in C++, has been seamlessly integrated into CUGR [5]. The experiments were conducted on a 64-bit Linux machine featuring a 12900k CPU and 128 GB of memory. The chosen cases were sourced from the ICCAD 2019 [6] competition. Despite CUGR's [5] successful mitigation of overflow issues in certain instances, a subset of larger-scale cases remained only partially addressed. Among the 20 competition test cases, three continued to exhibit overflow even after the application of CUGR [5], in addition to five undisclosed cases derived from the same competition. Consequently, we focused our experimentation on eight specific cases. Our results were systematically compared during practical testing with those obtained using CUGR [5].

We comprehensively compared the quality and runtime of our global router with the advanced global router CUGR [5], as detailed in Table I. In these benchmark tests, designs without metal5 utilized nine metal layers, while designs with metal5 were identical but incorporated five metal layers. The table presents five performance metrics, including scores (sum of wirelength, vias, and overflow), wirelength, vias, total overflow, and overall routing runtime (CUGR [5] + ours). Time is measured in seconds, and via count represents the total of wirelength and via count.

Our global router demonstrated a 0.9% improvement in total scores, a 0.5% reduction in wirelength, a 2.6% reduction in vias, and a substantial 17.1% decrease in overflow; particularly noteworthy is the case of $isp18\_test10\_metal5$, where we achieved almost a twofold reduction in overflow. It is important to note that although we experienced a 58.4% increase in runtime, this remains within reasonable bounds.

5 CONCLUSIONS

This paper introduces a P&R co-optimization engine designed to reduce congestion and bridge the gap between placement and routing. We integrate this engine into CUGR [5] for global routing optimization. In comparison to CUGR [5]'s experimental outcomes, our solution successfully mitigates overflow through the strategy of cell movement. Upon integrating our engine into the open-source code of CUGR [5], we observe slight improvements in total scores, wirelength, and via count, along with a 17.1% reduction in total overflow. Notably, the additional runtime remains within reasonable bounds. This underscores the capability of our method to enhance routing quality while maintaining acceptable performance levels.

ACKNOWLEDGMENTS

This Research is Supported by the National Key Research and Development Program from the Ministry of Science and Technology of the PRC (No.2021ZD0110600), Sichuan Provincial M. C. Integration Office Program, and IEDA Laboratory of SWUST.

REFERENCES

[1] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong Wang. 2018. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2018), 1717–1730.

[2] Kai-Shun Hu, Ming-Jen Yang, Tao-Chun Yu, and Guan-Chuen Chen. 2020. ICCAD-2020 CAD contest in routing with cell movement. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–4.

[3] Kai-Shun Hu, Tao-Chun Yu, Ming-Jen Yang, and Chin-Fang Cindy Shen. 2021. 2021 ICCAD CAD Contest Problem B: Routing with Cell Movement Advanced. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–5.

[4] Chau-Chin Huang, Hsin-Ying Lee, Bo-Qiao Lin, Sheng-Wei Yang, Chin-Hao Chang, Szu-To Chen, Yao-Wen Chang, Tung-Chieh Chen, and Ismail Bustany. 2017. 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 (2017), 669–681.

[5] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, and Evangeline FY Young. 2020. Cugr: Detailed-routability-driven 3d global routing with probabilistic resource model. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[6] Ulf Schlichtmann, Sabya Das, Ing-Chao Lin, and Mark Po-Hung Lin. 2019. Overview of 2019 CAD Contest at ICCAD. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–2. https://doi.org/10.1109/ICCAD45719.2019.8942133