A Fast and Robust Global Router with Capacity Reduction Techniques

Yun-Kai Fang, Ye-Chih Lin, Ting-Chi Wang
Department of Computer Science, National Tsing Hua University, Taiwan
ykfang@gapp.nthu.edu.tw, a0923327329@gmail.com, tcwang@cs.nthu.edu.tw

2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC) | 979-8-3503-9354-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASP-DAC58780.2024.10473789

***Abstract*—As design rules become more complex in new technology nodes, signal routing presents increasing challenges. In order to reduce the complexity, routing is typically divided into global routing (GR) and detailed routing (DR). However, a feasible GR solution may not always translate to a feasible DR solution due to resource mismatches between the two stages. In this work, we propose capacity reduction techniques that are applied in GR while aiming to enhance detailed-routability and bridge the mismatches between GR and DR. Encouraging experimental results demonstrate that after including the proposed capacity reduction techniques, the outdated open-source global router NTHU-Route 2.0 is revitalized and outperforms the state-of-the-art global router of TritonRoute-WXL. We achieve all DRC-clean solutions with 7.8% less via usage, 29% less non-preferred usage, and 2% DR quality score improvement while only increasing wirelength by 0.4%. Additionally, our GR runtime and DR runtime are respectively 44% and 31% shorter.**

I. Introduction

Signal routing is a challenging task in modern physical design flows. In order to reduce the complexity, it is typically divided into global routing (GR) and detailed routing (DR). GR partitions a design into a set of global cells (called Gcells) and selects a set of interconnected Gcells for each net as a routing guide for DR. DR finalizes the routing based on the GR results. Since DR uses the GR solution as a guide, the GR solution's quality significantly impacts the chip's final quality.

In a multi-layer global routing instance, each layer is partitioned into Gcells, and the boundary between two adjacent Gcells on the same layer is associated with a capacity, indicating the number of available routing tracks. The global routing problem is usually modeled on a 3D grid graph, where each Gcell represents a vertex, and each boundary or via signifies an edge. In this paper, we call a boundary edge on the same layer Gedge.

A. Previous Works

In 2019, the CAD Contest at ICCAD [1] organized the detailed-routability-driven global routing contest. This contest required global routers to create their own resource model, and the result is evaluated based on the detailed routing solution.

A global router named CUGR [2], the winner of the 2019 ICCAD contest, was developed based on the 2019 contest requirements. CUGR is a 3D router employing a probabilistic resource model, 3D pattern routing, and two-level 3D maze routing. Several subsequent works [3][4][5] focus on accelerating the different stages of CUGR using GPUs. However, these works do not aim to improve DR quality.

CUGR 2.0 [6] was proposed recently. It uses a data structure called routing DAG to abstract the pattern routing problem. The routing DAG can also be augmented to bypass the congested routing regions for some nets without performing maze routing. It also performs sparse maze routing to balance runtime and solution quality.

Another global router named SPRoute 2.0 [7] was also developed based on the 2019 contest. It introduces the soft capacity method, which utilizes pin density and Rectangular Uniform wire Density (RUDY) [8] to reserve space for detailed-routability. However, the soft capacity method considered in SPRoute 2.0 is relatively oversimplified and inconsiderate. The same equations are used to calculate soft capacity for different layers with varying parameters, which does not distinguish the pin density contributed by macros and the preferred routing direction of each layer. Also, the pin density in SPRoute 2.0 simply counts the number of pins in a Gcell while overlooking the resources around the Gcell.

An open-source unified global-detailed router, TritonRoute-WXL (TR-WXL) [9], has been proposed recently. Compared to the two global-detailed routing (GDR) flows, CUGR+Dr. CU [10] and SPRoute 2.0+Dr. CU flow, TR-WXL outperforms them and can provide solutions of 99.99% fewer DRC count. In this paper, we call the global router of TR-WXL TR-GR and the detailed router of TR-WXL TR-DR.

B. Our Motivation and Contributions

TR-WXL employs consistent routing information, such as pin access location and pin access layer, to bridge the gap between GR and DR stages. However, our experimental findings reveal that TR-WXL can still produce DRCs for more challenging testcases in the 2019 ICCAD contest benchmark suite, which were not reported in their paper. Furthermore, some testcases fail to converge in the DR stage within 24 hours. These issues indicate that TR-GR's resource modeling still has room for improvement. Inspired by SPRoute 2.0, which proposes reserving some resources during the GR stage to enhance detailed-routability, we aim to develop simple yet effective capacity reduction techniques for more accurate resource modeling.

The contributions of this paper are summarized as follows:

- We propose simple and effective capacity reduction techniques to boost the detailed-routability of a global router. Additionally, two patching techniques are added to improve the solution quality further.

979-8-3503-9354-5/24/$31.00 ©2024 IEEE

TABLE I: Evaluation metrics of DR.

Category	Metrics	Weight
Routing	Length of wire	0.5
	Number of vias	4
Non-Preferred Usage	Length of off-track wires	0.5
	Number of out-of-guide vias	1
	Length of off-track wires	1
	Number of off-track vias	1
	Length of wrong-way wire	1
DRC Violations	Number of min-area violations	500
	Number of spacing violations	500
	Number of short violations	500
	Area of metal/cut shorts	500

- Our capacity reduction techniques are easy to implement and have negligible runtime overhead.
- Experimental results show that the obsolete open-source global router NTHU-Route 2.0 [11] is revitalized with our capacity reduction techniques and outperforms the state-of-the-art GDR flow TR-WXL. We achieved all DRC-clean solutions and 7.8% less via usage, 29% less non-preferred usage, and 2% DR quality score improvement with only increasing wirelength by 0.4%. Additionally, our GR runtime and DR convergence time are respectively 44% and 31% shorter.
- To our best knowledge, this is the first work in academia that achieves all DRC-clean solutions in the benchmark suites of the 2019 ICCAD contest.

The rest of the paper is organized as follows: Section II gives the preliminaries. Section III presents our approach. Section IV reports and analyzes the experimental results. Finally, Section V concludes this paper.

II. Preliminaries

In this section, we first introduce the addressed problem. Then, since we use NTHU-Route 2.0 [11] to demonstrate our capacity reduction techniques, we will briefly review NTHU-Route 2.0. Finally, we will review the concept of RUDY [8] and the soft capacity method used in SPRoute 2.0 [7] as we will extend these techniques in our approach.

A. Problem Formulation

Traditionally, the objective of the global routing problem is to find an overflow-free tree for each net while minimizing the total wirelength and vias. In the 2019 ICCAD contest [1], a global routing solution is assessed based on the corresponding detailed routing result. The evaluation metrics are presented in Table I. The final DR quality score is the weighted sum of the metrics in Table I. In this paper, we follow the 2019 ICCAD contest metrics to evaluate routing results.

B. NTHU-Route 2.0

In this subsection, we briefly review NTHU-Route 2.0. Fig. 1, excluding the five new steps, illustrates the flow of NTHU-Route 2.0, which consists of four stages: the initial stage, main stage, refinement stage, and layer assignment stage. Note that those new steps are added by our approach and will be described in Section III.

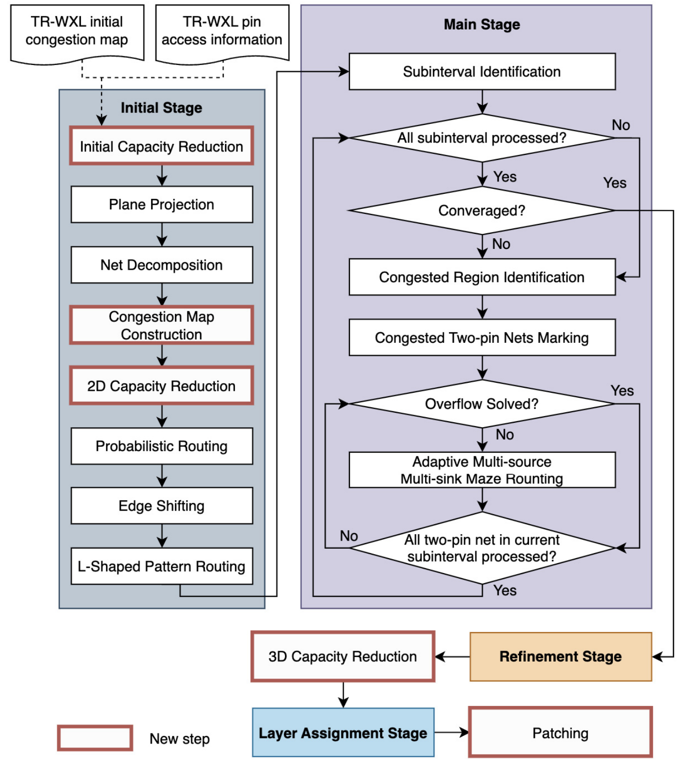

Fig. 1: Our flow.

The 2D routing process happens in the first three stages. Initially, NTHU-Route 2.0 projects a multi-layer design onto a plane, decomposes each multi-pin net into two-pin nets and then routes each net using two probabilistic L-shaped patterns. It adjusts the topology of each multi-pin net and reroutes each two-pin net with the lowest-cost L-shaped pattern. In the main stage, overflowed two-pin nets are iteratively ripped up and re-routed using monotonic routing. If no overflow-free path is found, maze routing is applied, and the cheaper cost path is accepted. The refinement stage focuses on identifying overflow-free paths, disregarding wirelength. Finally, a dynamic programming-based layer assignment method maps the 2D solution to the original layers, yielding a 3D routing solution.

C. RUDY

Rectangular Uniform Wire Density (RUDY) estimates wirelength density on a 2D routing grid. For each net, RUDY computes the sum of the width and height of the net's bounding box and evenly distributes the wirelength across all vertices of the 2D routing grid within the bounding box. Throughout this paper, we use the notation $Gcell(x, y)$ to represent the Gcell located at coordinates (x, y). The RUDY value contributed by a net n to $Gcell(x, y)$ is calculated using Equation (1).

$$RUDY_n(x,y) = \begin{cases} \dfrac{wirelength(n)}{bbox\ area(n)}, & \text{if } x, y \in bbox(n) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Here $bbox(n)$ is the net n's bounding box. Let N denote the set of nets. The RUDY value of $Gcell(x, y)$ is computed by summing up the RUDY value of all nets within $Gcell(x, y)$, which is shown in Equation (2).

$$RUDY(x, y) = \sum\nolimits_{n \in N} RUDY_n(x, y) \tag{2}$$

D. SPRoute 2.0

SPRoute 2.0 estimates congestion at $Gcell(x, y)$ by adding the pin density (i.e., the number of pins in $Gcell(x, y)$) to a constant w times the RUDY value of $Gcell(x, y)$, using Equation (3).

$$cong(x, y) = pin\_density(x, y) + w \times RUDY(x, y) \tag{3}$$

The soft capacity of an edge e in SPRoute 2.0 is calculated using Equation (4).

$$soft\_capacity(e) = ratio(cong(e)) \times hard\_cap(e) \tag{4}$$

In Equation (4), $hard\_cap(e)$ represents the capacity of e, and $cong(e)$ is the mean of congestion values of the two adjacent Gcells connected by e. The $ratio$ function is computed using Equation (5).

$$ratio(c) = min + \frac{max - min}{1 + exp((c - c_{mid}) \times k)} \tag{5}$$

Here, min and max represent the minimum and maximum values of the $ratio$ function. k controls the slope of the $ratio$ function, and c_{mid} is the midpoint of the $ratio$ function. The min, max, k, and c_{mid} are user-defined parameters.

III. Our Approach

In this section, we present our approach. We provide an overview of our approach in the first subsection, followed by detailed explanations in the subsequent subsections.

A. Overview

We modify NTHU-Route 2.0 [11] by integrating our capacity reduction and patching techniques. Specifically, we add the following steps *Initial Capacity Reduction*, *Congestion Map Construction*, *2D Capacity Reduction*, *3D Capacity Reduction*, and *Patching* into NTHU-Route 2.0. Since we will utilize TR-DR for detailed routing, we initially generate the initial congestion map and the pin access information, the same as those used in TR-GR, and use them as a part of the input to our router. Recall that TR-GR is the global router of TR-WXL, and TR-DR is the detailed router of TR-WXL. The initial congestion map provides information about the capacity and demand of each Gcell that takes available routing tracks and pin shapes or obstacles into account. On the other hand, the pin access information specifies the Gcell location of each pin in each net. The overall flow of our approach is illustrated in Fig. 1.

Since NTHU-Route 2.0 operates at the Gedge level, our approach begins by setting up the edge capacity and then applies capacity reduction within the *Initial Capacity Reduction* step. Following the *Net Decomposition* step, where FLUTE [12] is employed to break down multi-pin nets into two-pin nets, we construct five congestion maps during the *Congestion Map Construction* step. These five congestion maps are then utilized in both the *2D Capacity Reduction* and *3D Capacity Reduction* steps, which take place before the 2D global routing and layer assignment stages of NTHU-Route 2.0, respectively. Finally, the *Patching* step is added to the flow after the layer assignment stage to improve detailed-routability.

B. Initial Capacity Reduction

This subsection presents the process of setting up the initial edge capacity based on the Gcell-based initial congestion map and the criterion for reducing the capacity of an edge.

Since NTHU-Route 2.0 does not consider the edge capacity between layers, we only need to set the edge capacity within the same layer. An edge in a 3D routing grid graph with preferred routing directions connects only two adjacent Gcells; hence, we set the edge capacity as the minimum capacity between the two adjacent Gcells. Additionally, we set the capacity of each edge on Metal Layer 1 to zero to prevent wire routing on that layer.

Next, we describe the criterion for reducing the capacity of an edge. If at least one of the two adjacent Gcells connected by an edge has a demand greater than zero (i.e., there are obstacles inside these Gcells), we multiply the edge's capacity by a parameter $init\_discount \in [0.6, 1.0]$ otherwise the edge's capacity remains the same. The $init\_discount$ value remains the same for all the edges that need to reduce capacity.

This step addresses the resource mismatch between GR and DR around pin shapes or obstacles. It is straightforward to implement and incurs minimal runtime overhead. Despite its simplicity, it significantly impacts the quality of the DR results.

C. Congestion Map Construction

Before performing the 2D/3D capacity reduction, we generate five Gcell-based congestion maps: (1) *Cong_Map2D*, (2) *CongV_Map3D*, (3) *CongH_Map3D*, (4) *NonZeroCongV_Map3D*, and (5) *NonZeroCongH_Map3D*. The *Cong_Map2D* is utilized for 2D capacity reduction, while the other four maps are employed for 3D capacity reduction. All five maps are of the same size as the 2D grid graph.

For each map, we calculate the congestion value for each Gcell using the format of Equation (3) with the parameter w set to one. The congestion value in Equation (3) consists of two components: (1) a pin density term and (2) a RUDY term. Unlike SPRoute 2.0 [7], we calculate each term in two different ways. We will first explain the methods for computing these terms and then summarize how to use these terms to generate each congestion map.

Pin density term: The pin density term is calculated differently depending on whether we want to consider (a) all pins or (b) only non-zero pins. Non-zero pin refers to a pin on layers other than Metal Layer 1. (Note that the layer index of Metal Layer 1 equals zero in programming)

(a) Pin density: The pin density at $Gcell(x, y)$ is calculated using Equation (6).

$$pin\_density(x, y) = \frac{Number\ of\ pins\ in\ Gcell(x, y)}{cap(e^N_{x,y}) + cap(e^E_{x,y})} \quad (6)$$

In Equation (6), The $cap(e^N_{x,y})$/$cap(e^E_{x,y})$ represents the capacity of the north/east edge of $Gcell(x, y)$. Our pin density differs from that of SPRoute 2.0, which does not consider the capacity around a Gcell.

(b) Non-zero pin density: The non-zero pin density is calculated by counting the number of pins above Metal Layer 1. This calculation considers the pin density contributed by macros, which is not considered in SPRoute 2.0.

RUDY term: There are also two ways to calculate the RUDY term. Note that the RUDY term is calculated after decomposing multi-pin nets into two-pin nets, which better reflects the routing behavior of our global router.

(a) 2D RUDY: We use the same Equations (1) and (2) described in section II to compute *2D RUDY*.

(b) Horizontal/Vertical RUDY: To better consider the preferred routing direction of each layer, we propose a concept called *Horizontal/Vertical RUDY*. The $Horizontal\ RUDY_n$ contributed by each net n to $Gcell(x, y)$ is calculated by replacing the $wirelength(n)$ in Equation (1) with the width of net n's bounding box. Then, the *Horizontal RUDY* value of a Gcell is calculated by Equation (2) using $Horizontal\ RUDY_n$. *Vertical RUDY* can be similarly defined.

Table II summarizes which pin density and RUDY term are involved in calculating each congestion map. The maps *NonZeroCongV_Map3D* and *NonZeroCongH_Map3D* utilize the non-zero pin density for congestion value calculation, while the other maps employ the pin density. On the other hand, *CongV_Map3D* and *NonZeroCongV_Map3D* utilize *Vertical RUDY*, whereas *CongH_Map3D* and *NonZeroCongH_Map3D* employ *Horizontal RUDY* for congestion value calculation.

TABLE II: Summary of calculating congestion values for each map. A checkmark in the table represents which way a corresponding term is involved when calculating a congestion map.

	Pin density term		RUDY term	
	(a)	*(b)*	*(a)*	*(b)*
Cong_Map2D	✓		✓	
CongV_Map3D	✓			✓
CongH_Map3D	✓			✓
NonZeroCongV_Map3D		✓		✓
NonZeroCongH_Map3D		✓		✓

D. 2D Capacity Reduction

This subsection provides a detailed explanation of our 2D capacity reduction method. Before the 2D global routing stages of NTHU-Route 2.0, we use Equations (4) and (5) to reduce the edge capacity. The congestion of an edge e is the mean of congestion values of the two adjacent Gcells connected by e in *Cong_Map2D*. In Equation (5), we choose the average congestion $\overline{Cong\_Map2D}$ of each Gcell in *Cong_Map2D* as the midpoint of the $ratio$ function for normalization.

The *2D Capacity Reduction* technique forces a global router to avoid congested areas and thus improve detailed-routability. Note that the *2D Capacity Reduction* step explicitly targets the capacity reduction of 2D edges, which are exclusively used in the 2D routing stage. The original 3D edge capacities, determined after the *Initial Capacity Reduction* step, remain unchanged.

E. 3D Capacity Reduction

This subsection presents our method for 3D capacity reduction. The *3D Capacity Reduction* step is applied before the layer assignment stage. To account for the resources occupied by local nets in each Gcell, we first calculate the original capacity for each Gcell in each layer by summing up the capacities of the two adjacent edges. Note that the edge capacity considered here is the one after the *Initial Capacity Reduction* step.

Next, we utilize the four congestion maps *CongV_Map3D*, *CongH_Map3D*, *NonZeroCongV_Map3D*, and *NonZeroCongH_Map3D* to calculate a reservation ratio for each Gcell and each map. For instance, Equation (7) illustrates how to calculate the reservation ratio $ratioV(x, y)$ for $Gcell(x, y)$ using the congestion value $congV(x, y)$ (= $CongV\_Map3D(x, y)$).

$$ratioV(x, y) = min + \frac{max - min}{1 + exp((congV(x, y) - \overline{CongV\_Map3D}) \times k)} \quad (7)$$

In Equation (7), $\overline{CongV\_Map3D}$ is the average congestion of each Gcell in *CongV_Map3D*, while min, max, and k are user-defined parameters and have the same definitions as in Equation (5). Similarly, we can derive the other reservation ratios, namely $ratioH(x, y)$, $ratioNonZeroV(x, y)$, and $ratioNonZeroH(x, y)$ for each $Gcell(x, y)$.

In our *3D Capacity Reduction* step, we utilize the reservation ratios $ratioV$ and $ratioH$ to determine the capacity reduction for Gcells located under Metal Layer 4. (We use Metal Layer 4 as a boundary since the pins of macros are mainly located above Metal Layer 4 in our benchmark suites.) For instance, if the preferred routing direction of Metal Layer 2 is horizontal, we reduce the capacity of $Gcell(x, y)$ by $\alpha\%$ of its original capacity if $ratioH(x, y)$ is less than r_1. Otherwise, no capacity reduction is applied to $Gcell(x, y)$. Similarly, we use the reservation ratios $ratioNonZeroV$ and $ratioNonZeroH$ for Gcells located above or on Metal Layer 4 to adjust their capacities. For instance, if the preferred routing direction of Metal Layer 5 is vertical, we reduce the capacity of $Gcell(x, y)$ by $\beta\%$ of its original capacity if $ratioNonZeroV(x, y)$ is less than r_2. Otherwise, no capacity reduction is performed on $Gcell(x, y)$. The parameters α, β,

TABLE III: Comparison of DR quality score, GR runtime, and DR convergence time between TR-WXL and our GDR flow.

	Quality Score		GR Runtime (s)		DR Runtime (s)	
	Ours	TR-WXL	Ours	TR-WXL	Ours	TR-WXL
ispd18_test5	**15,534,187**	15,964,387	**12**	37	**101**	125
ispd18_test8	**37,002,876**	38,171,902	**30**	139	**425**	442
ispd18_test10	**39,870,651**	41,210,590	**36**	120	**318**	350
ispd19_test7	**79,205,086**	79,712,143	**59**	109	723	**697**
ispd19_test8	**121,191,069**	122,118,761	**74**	167	1,055	**890**
ispd19_test9	**187,512,156**	189,307,818	**122**	284	1,769	**1,577**
ispd18_test5_metal5	**15,507,642**	15,688,396	168	**85**	**136**	200
ispd18_test8_metal5	**36,240,355**	37,247,213	**355**	367	**358**	389
ispd18_test10_metal5	**40,180,522**	-	**92**	1,242	**405**	$> 24hrs$
ispd19_test7_metal5	**73,467,854**	-	**879**	1,418	**797**	$> 24hrs$
ispd19_test8_metal5	**117,943,991**	120,179,249	1,023	**457**	**1,221**	13,062
ispd19_test9_metal5	**181,925,297**	184,712,356	1,118	**453**	**1,722**	4,192
GEO. Mean Ratio	**0.98**	1.00	**0.56**	1.00	**0.69**	1.00

r_1, and r_2 are user-defined parameters and are set to 30, 20, 0.6, and 0.65, respectively, in our implementation and are consistent among all benchmarks.

Finally, we reduce the capacity of the Gcells that contain local nets by γ units, where γ is set to two in our experiment. Once this reduction is performed on all relevant Gcells, we use the Gcell capacities to calculate Gedge capacities similar to the method described earlier in section III-E. This calculation is necessary because the layer assignment algorithm [13] of NTHU-Route 2.0 operates at the Gedge level.

The layer assignment algorithm adopted by NTHU-Route 2.0 does not consider via overflow; however, our *3D Capacity Reduction* technique can automatically avoid assigning too many demands on an edge where its neighboring Gcells cannot accommodate too many vias. This reduction technique provides a simple way to remedy the algorithm's shortcomings and thus prevent via overflow to achieve better detailed-routability. Moreover, thanks to the *3D Capacity Reduction* technique, our global router does not need to perform the 3D ripup-and-reroute step after layer assignment while TR-GR is still needed to escape congested areas.

F. Patching

We adopt two patching techniques similar to those proposed in CUGR [2]. The patching techniques add some Gcells in the global routing result of a net. The first technique patches Gcells around a pin whose neighboring Gcells at the upper or lower layer lack resources to access the pin. The second patches Gcells for the Gcell with insufficient resources on a long wire segment to offer more flexibility to the detailed router to switch the layer for routing.

IV. Experimental Results

We implemented our approach in C++ on a Linux workstation with an AMD EPYC 7543 2.8 GHz CPU and 256 GB memory. We obtained the source code of NTHU-Route 2.0 [11] from [14]. Our experiments used benchmark suites from the 2019 ICCAD contest [1] in 32-nm technology nodes. These benchmarks comprise designs with up to 899K standard cells and 895K nets. The benchmarks are categorized into two groups: those without the *_metal5* suffix, which represent designs with nine metal layers, and those with the *_metal5* suffix, which are the same designs with only five metal layers and I/O pins redistributed primarily on the lower metal layers. The testcases with *_metal5* suffix, not reported in the paper of TR-WXL [9], present a more challenging scenario due to their limited routing resources. In our experiments, we also fine-tuned the user-defined parameters, as mentioned in the previous section, accounting for the nature of each benchmark.

We compare NTHU-Route 2.0 enhanced with our capacity reduction and patching techniques with TR-GR since TR-GR with its detailed router (TR-DR) achieves the best results among all academic routers. In contrast to the 2019 ICCAD contest, where Dr. CU [10] was used for detailed routing, we employed TR-DR in our experiments. This choice allows for a direct comparison between our approach and TR-WXL's global routing. We utilized the same initial congestion map and pin access information to ensure a fair comparison between our GDR flow and TR-WXL. To evaluate the quality of the detailed routing, we used Cadence Innovus [15], which follows the same evaluation metrics used in the 2019 ICCAD contest. The evaluation metrics are summarized in Table I, where a lower score signifies a better performance.

A. Comparison to TR-GR

Table III compares our GDR flow and TR-WXL in terms of DR quality score, GR runtime, and DR runtime. Our routing flow utilizes a single thread for global routing, whereas TR-GR employs **8** threads. The DR runtime represents the time taken to converge based on the routing guide provided by the global routing stage. For detailed routing, we utilize 128 threads. Note that each "-" in Table III and IV means TR-DR failed to converge within 24 hours.

According to Table III, our GDR flow achieves a 2% improvement in quality score. Regarding runtime, our GR runtime and DR runtime are respectively 44% and 31% shorter. Furthermore, in two cases, namely $ispd18\_test10\_metal5$ and $ispd19\_test7\_metal5$, our GDR flow successfully achieves

TABLE IV: Comparison of (unweighted) DR metrics between TR-WXL and our GDR flow.

	Wirelength		#Vias		Non-Preferred Usage		#Shorts		#Min-Area & #Spacing	
	Ours	TR-WXL	Ours	TR-WXL	Ours	TR-WXL	Ours	TR-WXL	Ours	TR-WXL
ispd18_test5	**27,357,147**	27,396,599	**831,772**	897,844	**201,389**	480,738	0	0	0	0
ispd18_test8	64,621,104	**64,508,509**	**2,009,389**	2,362,426	**704,776**	1,228,825	0	0	0	0
ispd18_test10	**68,242,326**	68,266,423	**2,191,240**	2,565,687	**1,406,735**	2,000,825	0	0	0	0
ispd19_test7	120,527,054	**120,429,939**	**3,886,346**	3,896,349	**3,481,789**	4,010,955	0	0	**0**	1
ispd19_test8	185,592,132	**184,940,092**	**6,062,030**	6,425,825	4,265,573	**4,060,109**	0	0	0	0
ispd19_test9	280,394,296	**279,643,408**	**10,288,792**	10,673,288	**6,355,762**	6,989,186	0	0	0	0
ispd18_test5_metal5	27,200,889	**26,742,896**	**811,864**	903,527	**293,755**	510,987	**0**	10	**0**	4
ispd18_test8_metal5	62,720,998	**62,277,768**	**2,005,243**	2,293,936	**897,638**	1,541,673	**0**	8	**0**	8
ispd18_test10_metal5	**68,228,450**	-	**2,204,242**	-	**1,693,057**	-	**0**	-	**0**	-
ispd19_test7_metal5	**109,079,978**	-	**3,930,544**	-	**3,285,167**	-	**0**	-	**0**	-
ispd19_test8_metal5	179,090,690	**176,898,224**	**6,097,064**	6,346,347	**4,122,523**	5,375,573	**0**	817	**0**	29
ispd19_test9_metal5	**269,232,863**	269,577,838	**10,078,824**	10,458,213	**7,177,484**	7,984,121	**0**	229	**0**	4
GEO. Mean Ratio	1.004	**1.000**	**0.922**	1.000	**0.710**	1.000	-	-	-	-

convergence in the DR stage, while TR-WXL fails to terminate within 24 hours.

Table IV provides a detailed breakdown of the detailed routing results achieved by our router compared to TR-GR. The table shows that our GDR flow achieves a 7.8% reduction in vias and a 29% decrease in non-preferred usage while experiencing only a slight increase of 0.4% in wirelength.

Of utmost significance is the fact that our GDR flow delivers all DRC-clean solutions. In contrast, the solutions provided by TR-WXL exhibit DRC issues in five cases. Moreover, there are two benchmarks where TR-WXL failed to terminate within 24 hours. These results highlight the superior performance of our GDR flow in terms of solution quality.

Here, we want to emphasize that even without the patching techniques, our GDR flow can still deliver all DRC-clean solutions and have faster runtime in both GR and DR stages than TR-WXL but will increase non-preferred usage. We omit the details to save space.

B. Comparison to Other Global-Detailed Routing Flows

We also compare our GDR flow to CUGR 2.0+Dr. CU and SPRoute 2.0+Dr. CU with scores quoted from their papers [6] [7]. For CUGR 2.0+Dr. CU, we achieve a 5.2% reduction in the DR quality score and a 2.2% decrease in the score of wirelength plus via. For SPRoute 2.0+Dr. CU, we achieve an 8% reduction in the DR quality score and a 4.7% decrease in the score of wirelength plus via. Both non-preferred usage scores are almost twice as large as those two flows; however, we achieve all DRC-clean solutions while those have huge DRCs.

V. Conclusion

This paper proposes simple yet effective capacity reduction techniques to address the resource mismatches between GR and DR. They are easy to implement with negligible runtime overhead. Experimental results show that the outdated NTHU-Route 2.0 [11] with our techniques outperforms the state-of-the-art TritonRoute-WXL [9]. To our best knowledge, this is the first work in academia that achieves all DRC-clean solutions in the benchmark suites of the 2019 ICCAD contest. We believe our capacity reduction techniques can also be applied to other global routers, while the efficacy requires further study.

References

[1] S. Dolgov, A. Volkov, L. Wang and B. Xu, "2019 CAD Contest: LEF/DEF Based Global Routing," in *Proceedings of International Conference on Computer-Aided Design*, 2019, pp. 1-4.

[2] J. Liu, C.-W. Pui, F. Wang, and E. F. Y. Young, "CUGR: Detailed-Routability-Driven 3D Global Routing with Probabilistic Resource Model," in *Proceedings of Design Automation Conference*, 2020, pp. 1-6.

[3] S. Liu et al., "FastGR: Global Routing on CPU-GPU with Heterogeneous Task Graph Scheduler," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, 2022, pp. 760-765.

[4] S. Lin, J. Liu, E. F. Y. Young, and M. D. F. Wong, "GAMER: GPU-Accelerated Maze Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 583-593, 2023.

[5] S. Lin and M. D. F. Wong, "Superfast Full-Scale GPU-Accelerated Global Routing," in *Proceedings of International Conference On Computer-Aided Design*, 2022, pp. 1-8.

[6] J. Liu and E. F. Y. Young, "EDGE: Efficient DAG-based Global Routing Engine," in *Proceedings of Design Automation Conference*, 2023.

[7] J. He, U. Agarwal, Y. Yang, R. Manohar, and K. Pingali, "SPRoute 2.0: A detailed-routability-driven deterministic parallel global router with soft capacity," in *Proceedings of Asia and South Pacific Design Automation Conference*, 2022, pp. 586-591.

[8] P. Spindler and F. M. Johannes, "Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1-6.

[9] A. B. Kahng, L. Wang and B. Xu, "TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1076-1089, 2022.

[10] H. Li, G. Chen, B. Jiang, J. Chen and E. F. Y. Young, "Dr. CU 2.0: A Scalable Detailed Routing Framework with Correct-by-Construction Design Rule Satisfaction," in *Proceedings of International Conference on Computer-Aided Design*, 2019, pp. 1-7.

[11] Y.-J. Chang, Y.-T. Lee and T.-C. Wang, "NTHU-Route 2.0: A fast and stable global router," in *Proceedings of International Conference on Computer-Aided Design*, 2008, pp. 338-343.

[12] C. Chu and Y.-C. 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*, vol. 27, no. 1, pp. 70-83, 2008.

[13] T.-H. Lee and T.-C. Wang, "Congestion-Constrained Layer Assignment for Via Minimization in Global Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1643-1656, 2008.

[14] "NTHU Route 2.0." http://www.cs.nthu.edu.tw/~tcwang/nthuroute/

[15] "Cadence innovus implementation system." http://www.cadence.com