Hindawi
Complexity
Volume 2021, Article ID 8843271, 12 pages
https://doi.org/10.1155/2021/8843271

Research Article

Topology-Aware Bus Routing in Complex Networks of Very-Large-Scale Integration with Nonuniform Track Configurations and Obstacles

Ziran Zhu,[1] Zhipeng Huang,[2] Jianli Chen,[3] and Longkun Guo [4]

[1]*National ASIC System Engineering Research Center, Southeast University, Nanjing, China*
[2]*Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou, China*
[3]*State Key Laboratory of ASIC and System, Fudan University, Shanghai, China*
[4]*Shandong Key Laboratory of Computer Networks, School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan, China*

Correspondence should be addressed to Longkun Guo; longkun.guo@gmail.com

Received 9 September 2020; Revised 21 January 2021; Accepted 26 March 2021; Published 15 April 2021

Academic Editor: Shi Cheng

Copyright © 2021 Ziran Zhu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As one of the most important routing problems in the complex network within a very-large-scale integration (VLSI) circuit, bus routing has become much more challenging when witnessing the advanced technology node enters the deep nanometer era because all bus bits need to be routed with the same routing topology in the context. In particular, the nonuniform routing track configuration and obstacles bring the largest difficulty for maintaining the same topology for all bus bits. In this paper, we first present a track handling technique to unify the nonuniform routing track configuration with obstacles. Then, we formulate the topology-aware single bus routing as an unsplittable flow problem (UFP), which is integrated into a negotiation-based global routing to determine the desired routing regions for each bus. A topology-aware track assignment is also presented to allocate the tracks to each segment of buses under the guidance of the global routing result. Finally, a detailed routing scheme is proposed to connect the segments of each bus. We evaluate our routing result with the benchmark suite of the 2018 CAD Contest. Compared with the top-3 state-of-the-art methods, experimental results show that our proposed algorithm achieves the best overall score regarding specified time limitations.

1. Introduction

As the advanced technology node enters the deep nanometer era, routing has become much challenging because of the enormously growing scale of the large scale of very-large-scale integration (VLSI) circuit [1]. Among the routing problems, bus routing is attracting most research interest and has met new challenges: (1) all bits in each bus must be routed with the same routing topology; (2) nonuniform and complex routing track configurations; and (3) we need to handle obstacles. Particularly, the constraint according to which all bits belonging to the same bus must be routed following the same routing topology makes the previous routers not applicable to current topology-matching bus routing.

Previous works on bus routing focused mainly on printed circuit board (PCB) designs. For example, Tian and Watanabe [2] considered the delay-matching constraint in bus routing to meet several timing specifications. Yan and Wong [3] and Zhang et al. [4] handled the length-matching bus routing such that the wire lengths of all nets on the same bus are within the specified range. However, none of these works considered the constraint of maintaining the same topology for all bits on the same bus. Therefore, it is desirable to develop an effective and efficient topology-matching bus routing algorithm.

According to the previous criteria, the bits in a bus are considered to have the same topology if the following four criteria are met: (1) All bits have the same number of wires. (2) All wires traced from all bits have the same layer

sequencing. (3) All wires traced from all bits route towards the same direction. (4) Within each segment (a segment is a set of wires of different bits with the same sequence when traced from a set of pin shapes), the wires of different bits maintain the same or the reverse order as the order seen from the pin shapes.

Figure 1(a) illustrates a bus being routed successfully with the same topology. Supposing that we start tracking the wires from the pin shapes on the left, as shown in this figure, all wires traced from all bits have the same direction (rightward, downward, and rightward, resp.) and the same layer sequencing (L1, L2, and L1). In addition, within all segments (seg1, seg2, and seg3), the wires of different bits maintain the same order or the reverse order as the order of pin shapes.

Routing tracks are designed to help routers comply with various design requirements and help mask coloring, which are essential in advanced technology node [5]. Each routing track has a width constraint that only wires with width no larger than the constraint are allowed to be routed on the track. Since the routing requirements of different buses may be different (e.g., different wire widths and different wire spacing), the routing track configuration may be nonuniform. For example, Figure 1(b) shows five tracks that can be classified into two types (blue and green, resp.) based on the width constraints. The blue tracks have a larger width constraint and cover the whole design from bottom to top, while the green tracks have a smaller width constraint and only cover part of the design. Particularly, routing tracks can overlap with each other, and the distribution may be uneven. Such nonuniform routing track configuration imposes a great challenge to the bus routing.

Obstacles such as circuit components and power vias make bus routing even more challenging. Since such obstacles are scattered throughout certain layers, it is not possible to find continuous routable area if bus bits are not allowed to route between some of them.

Due to the high complexity of the routing problems, the routing process is typically divided into global routing, track assignment, and detailed routing. In global routing, the routing region is divided into coarse-grained grid cells (called g-cells), and rough routing regions are determined for each net through the connection between the g-cells. Next, track assignment allocates routing tracks to iroutes that are extracted from the global routing result. Finally, detailed routing finds a path for each net to connect the iroute and pins and completes the final routing.

In this paper, we propose an effective algorithm to solve the topology-matching bus routing problem considering obstacles and nonuniform track configurations. The major contributions of our work are summarized as follows:

(i) A track handling technique is presented to unify the nonuniform routing track configuration with obstacles.

(ii) We formulate the topology-aware single bus routing as an unsplittable flow problem (UFP), which is integrated into a negotiation-based global routing to determine the desired routing regions for each bus.

(iii) Under the guidance of the global routing result, a topology-aware track assignment is proposed to allocate tracks to each segment of buses, which significantly reduces the difficulty of maintaining the same routing topology in the subsequent steps.

(iv) A detailed routing scheme considering routing topology for all bus bits is presented to connect the segments of each bus.

(v) Experimental results show that our proposed bus routing algorithm is effective. Compared with the top 3 teams of the CAD Contest at ICCAD on Obstacle-Aware On-Track Bus Routing [5], our proposed algorithm can achieve the best overall score within the specified time.

The rest of this paper is organized as follows. Section 2 first introduces the bus routing preference metrics and then gives the problem statement. Section 3 details our bus routing algorithm. Section 4 provides the experimental results. Finally, conclusions are drawn in Section 5.

2. Preliminaries

In this section, we first introduce the bus routing preference metrics considered in the 2018 CAD Contest at ICCAD [5] and then formulate the topology-matching bus routing problem.

2.1. Bus Routing Preference Metrics. For successfully routed buses, the metrics such as wire length, the number of segments, the width of each segment, and the number of spacing violations are used to measure the bus routing quality in the contest [5]. We detail these four metrics as follows.

2.1.1. Wire Length. Wire length is a basic metric of routing quality. Longer wire lengths typically imply larger delays and larger power consumption [6]; hence routers are expected to minimize the wire length. The wire length of a bus is calculated by summing up the wire length of all bits, and taking detour will cause an increase in wire length.

2.1.2. Number of Segments. Since each layer has a preferred routing direction (either horizontal or vertical), more segments indicate that more vias are used. However, vias are undesirable due to their negative impacts on signal integrity, delay, routing area, and manufacturing yields [6]. Therefore, an ideal bus router should minimize the number of segments.

2.1.3. Width of Segments. If the direction of a segment is horizontal, then the width of the segment is defined as the y-coordinate of the topmost wire in the segment minus the y-coordinate of the bottommost wire in the segment; in contrast, if the direction of a segment is vertical, the

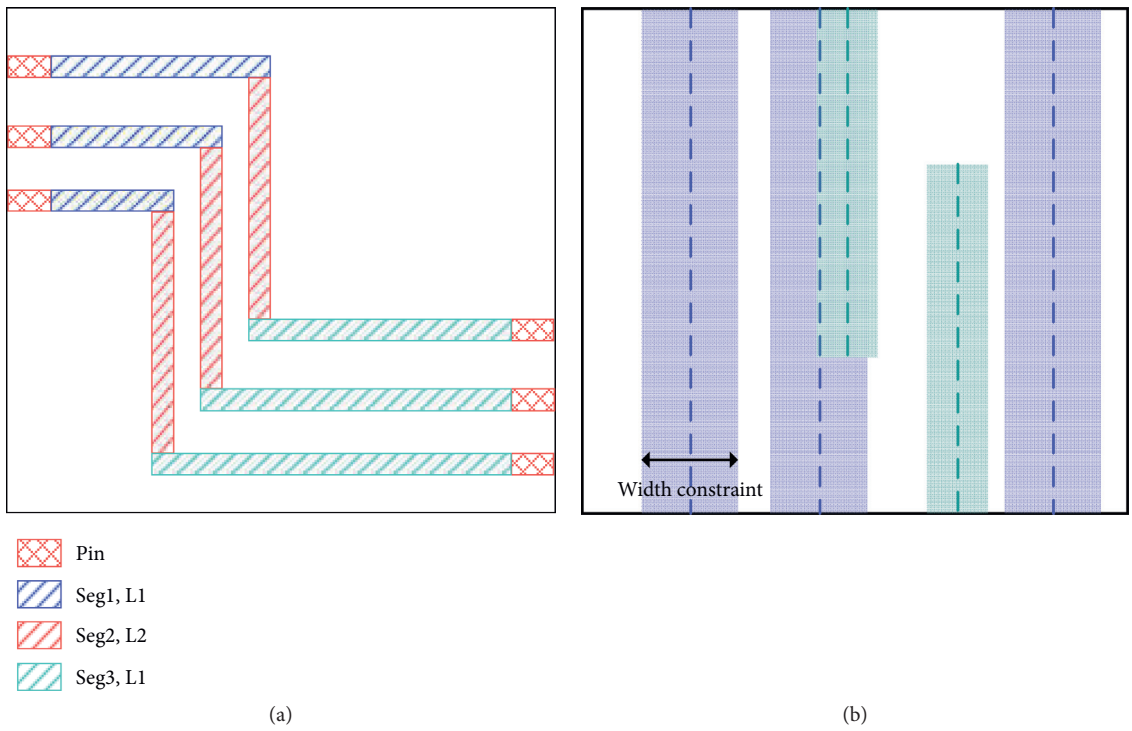

Figure 1: (a) A bus with the same routing topology. (b) Nonuniform routing track configuration.

calculation is between the x-coordinate of the rightmost and the leftmost wire [5]. The smaller the width of a segment is, the more compact the corresponding bus is.

2.1.4. Spacing Violation. Each layer has a spacing constraint which specifies the minimum distance that should be maintained between the routing paths of a bit and the design boundary, the obstacles, and other routing wires on that layer. Spacing violation will result in an additional penalty in the evaluation score.

Figure 2 shows an example of bus routing preference metrics, where Figure 2(a) gives an inferior routing result with a longer wire length, a large number of segments, and a larger width of segments, while Figure 2(b) gives a desired solution because its wire length and number of segments are minimized, and the width of segments is also smaller.

2.2. Problem Statement. We are given the following: (1) A design with l routing layers $\mathscr{L} = \{L_1, L_2, \ldots, L_l\}$. Each layer has a routing direction and a spacing constraint. The routing direction is either vertical or horizontal, and the spacing constraint specifies the minimum distance that should be maintained between the output routing path and the design boundary, the obstacles, and other routing wires in that layer. (2) A set of m routing tracks $\mathscr{T} = \{T_1, T_2, \ldots, T_m\}$. Each track T_i is represented by a line in a layer with a width constraint WT_i. The direction of each track is always the same as the routing direction of the layer that the track is on, and the width constraint requires that only the wires with a width smaller than or equal to the width constraint can be routed on the track. (3) A set of p obstacles $\mathscr{O} = \{O_1, O_2, \ldots, O_p\}$, which are scattered throughout certain layers. (4) A set of n buses $\mathscr{B} = \{B_1, B_2, \ldots, B_n\}$. Each bus B_i consists of NB_i bits, all bits have NP_i pin shapes, and the width constraint WB_{ij} $(1 \le j \le l)$ indicates that the routing paths for the bus B_i in layer L_j have the width equal to the width constraint.

The goal of topology-matching bus routing is to achieve as many successfully routed buses as possible, and the following metrics should also be minimized simultaneously: (1) the total wire length of all buses; (2) the number of segments; (3) compactness of each bus (i.e., the width of segments); (4) the number of spacing violations.

A bus is routed successfully if the following three hard constraints are met: (1) Routing paths of each bit connect all pin shapes of the bit and do not overlap with the paths of other bits. (2) All wires are on-track without violating the width constraint of the tracks and do not overlap with obstacles. (3) All bits are routed with the same topology.

3. Our Algorithm

The overall flow of our proposed algorithm is summarized in Figure 3, which mainly consists of four parts: (1) preprocessing, (2) global routing, (3) track assignment, and (4) detailed routing.

The preprocessing stage unifies the nonuniform routing track configuration with obstacles to support efficient query and simplify subsequent routing operations. In the global routing stage, we formulate the topology-aware single bus routing as UFP and integrate it into a negotiation-based global routing to determine the desired routing regions for each bus. The complexity of subsequent steps can be reduced

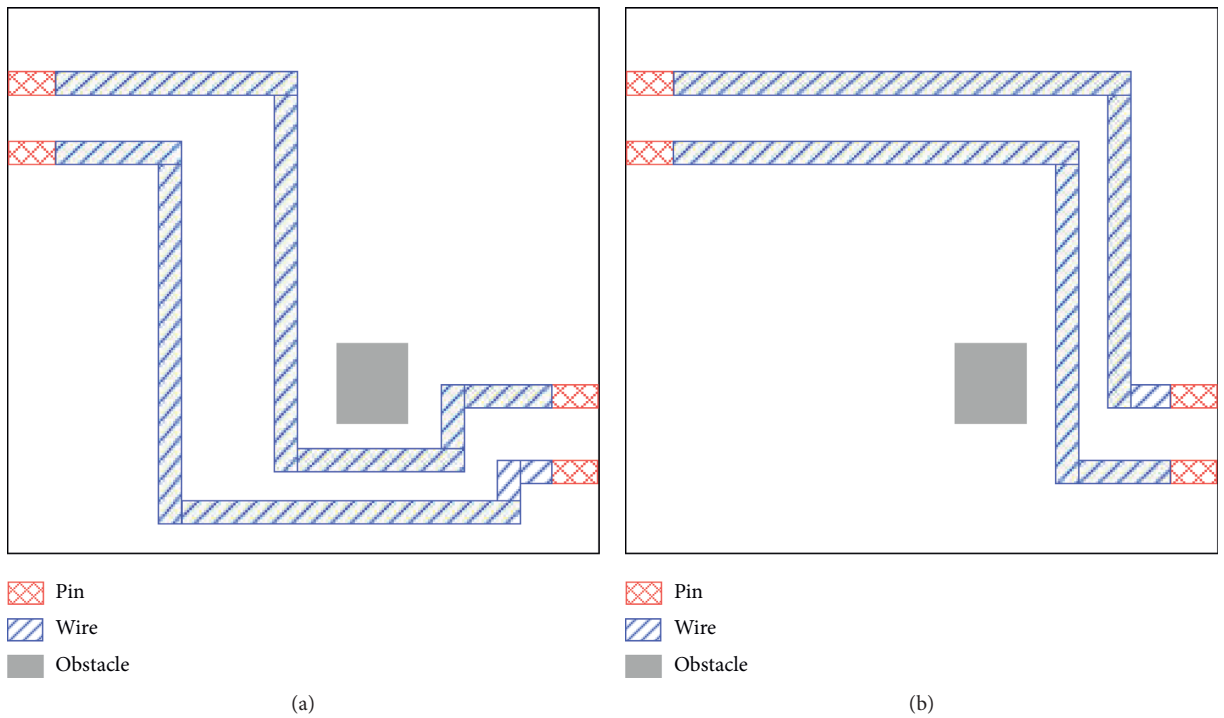

Figure 2: Example of bus routing preference metrics. (a) An inferior routing result. (b) A desired routing result.

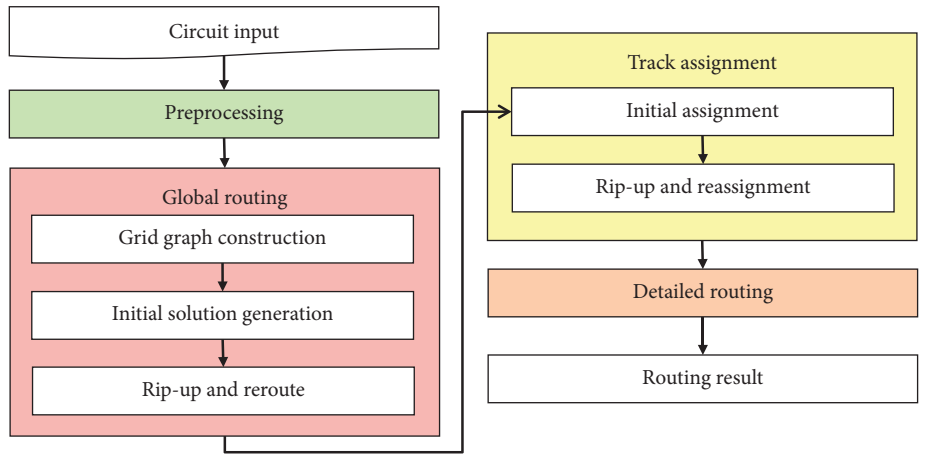

Figure 3: Our algorithm flow.

by confining its search space to the regions identified by the global routing stage. Under the guidance of the global routing result, the track assignment stage allocates tracks to each segment of buses, which significantly reduces the difficulty of maintaining the same routing topology in the subsequent step. Finally, the detailed routing stage connects the segments of each bus bit and obtains the final routing result. We shall detail these four major parts in the following subsections.

3.1. Preprocessing. In this subsection, we perform some preprocessing on the input data to simplify subsequent routing operations. Since the nonuniform routing track configuration and obstacles make the bus routing much more complicated, we first present a track handling technique to unify the track configuration while considering obstacles. Specifically, we treat each routing track uniformly as covering the entire design from top to bottom (or left to right), and each track has a set of intervals for recording the subtracks that have been used. Furthermore, if the centerlines of two tracks overlap, we will shrink or delete the used intervals of the track that has a smaller width.

Figure 4 shows three nonuniform routing tracks and an obstacle. In the figure, we treat these three tracks as covering the entire design from top to bottom, and tracks T_1, T_2, and T_3 have the sets of used intervals $\{I_1, I_2\}$, $\{I_4\}$, and $\{I_3\}$,

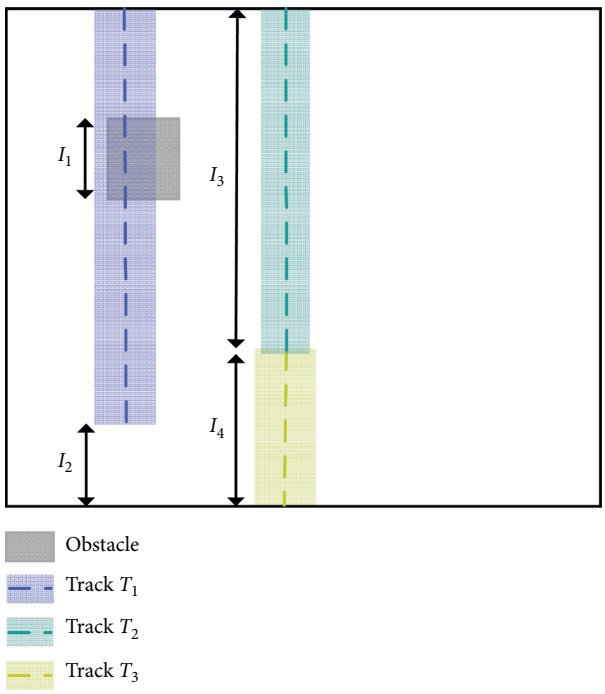

Figure 4: Example of our track handling technique.

respectively. The interval I_1 is occupied by the obstacle, and intervals I_2, I_3, I_4 are not covered by the corresponding tracks. In addition, since the centerlines of tracks T_2 and T_3 overlap and the track T_2 has a smaller width, we update the set of used intervals of track T_2 by deleting the interval I_4. Then the set of used intervals of track T_2 is finally empty, and thus the wires can go through directly from track T_2 to track T_3.

Besides, we adopt a minimum spanning tree algorithm to decompose each multipin bit into a set of two-pin bits and determine a preferred direction (horizontal or vertical) for each pin. That is, if the physical locations of the same pin shapes in different bits are horizontally distributed, then the preferred directions of the pin shapes are set as vertical; in contrast, if the physical positions of the same pin shapes in different bits are vertically distributed, then the preferred directions of the pin shapes are horizontal. The preferred direction of a pin is the desired direction of the wire that connects to the pin. For example, the preferred directions of all six pins in Figure 1(a) are horizontal, since the physical positions of the same pin shapes in different bits are vertically distributed.

3.2. Global Routing. To determine the desired routing regions for each bus and reduce the complexity of subsequent detailed routing, we formulate the topology-aware single bus routing as UFP and integrate it into a negotiation-based global routing scheme. The three main steps of our global routing scheme are elaborated as follows.

3.2.1. Grid Graph Construction. In the global routing stage, each routing layer is partitioned into a set of global cells (g-cells) as shown in Figure 5(a), and a corresponding grid graph can be constructed as shown in Figure 5(b). In the grid graph, each vertex represents a g-cell and each routing edge represents a boundary between adjacent g-cells, and any two adjacent layers are connected by vias. In addition, the number on each routing edge of Figure 5(b) indicates the capacity of the edge, which corresponds to the number of routing tracks that can be contained across the edge. Since solving the 3D global routing problem directly is time-consuming, we further project a multilayered design onto the 2D plane, and then a capacitated graph $G(V, E, u)$ is constructed.

Besides, each pin corresponds to a g-cell. If two pins of any two bits are in the same g-cells, then we temporarily combine the two bits as a bit. In this way, we can reduce the number of bits in each bus greatly in global routing, and the runtime of the global routing stage will be reduced. Take Figure 5(a) for example. There are two bits in a bus and each bit has two pins. Since the two pins of two bits are in the same g-cells, we combine the two bits as a bit, and the demand (number of tracks consumed) of each routed wire in the merged bit is 2.

3.2.2. Initial Solution Generation. Too many bends of a routing path not only increase the number of vias, resulting in poor routing quality, but also make it more difficult to maintain the same routing topology due to the increase of the number of segments. Therefore, we limit the number of bends in this initial solution generation step. The bits can be categorized into two types based on the preferred directions of pins. Figure 6(a) shows the pins of four bits from different buses with the same preferred direction, and the path connecting each bit has an even number of bends. Conversely, the pins of three bits from different buses shown in Figure 6(b) have orthogonally preferred directions, and the path connecting the two pins of each bit has an odd number of bends. Further, since we set a preferred direction for each pin, a bit has at most one path with one or zero bends, and we only need to determine $n-1$ $(n \geq 2)$ bending points in turn to obtain a path with n bends.

In this step, the number of bends of a path connecting each bit is limited to four. For each bus, let d_i represent the demand of bit i, $\mathscr{P}_i$ denote the set of paths for bit i, and $\mathscr{P}_i^T$ represent the set of paths with the same routing topology T in $\mathscr{P}_i$. Note that, the topology in global routing ensures that (1) all bits have the same number of wires and (2) all wires traced from all bits route towards the same direction, while temporarily ignoring the relative order of the wire of different bits. In addition, for each $P \in \mathscr{P}_i$, we have a non-negative variable $x(P)$ and a weight $w(P)$ associated with it. The weight $w(P)$ of path P is the sum of the weights of all the edges on the path, and the weight of edge e is defined as

$$w(e) = \frac{1}{1 + e^{(d(e)-u(e))}}, \tag{1}$$

where $d(e)$ represents the sum of the demands of the bits passing through e and $u(e)$ is the capacity of the edge e. For each edge e, the demand $d(e)$ is initialized to 0 and is updated once a bus is routed successfully. The weight $w(e)$

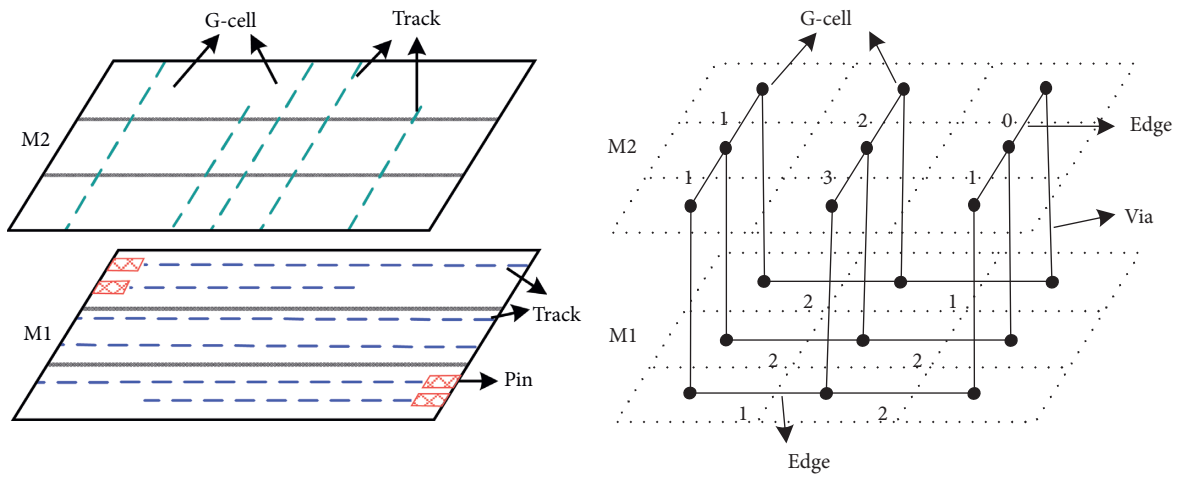

Figure 5: Example of grid graph construction. (a) Each routing layer is partitioned into a set of global cells (g-cells). (b) Each g-cell is modeled as a vertex, and abutting g-cells are connected by routing edges.

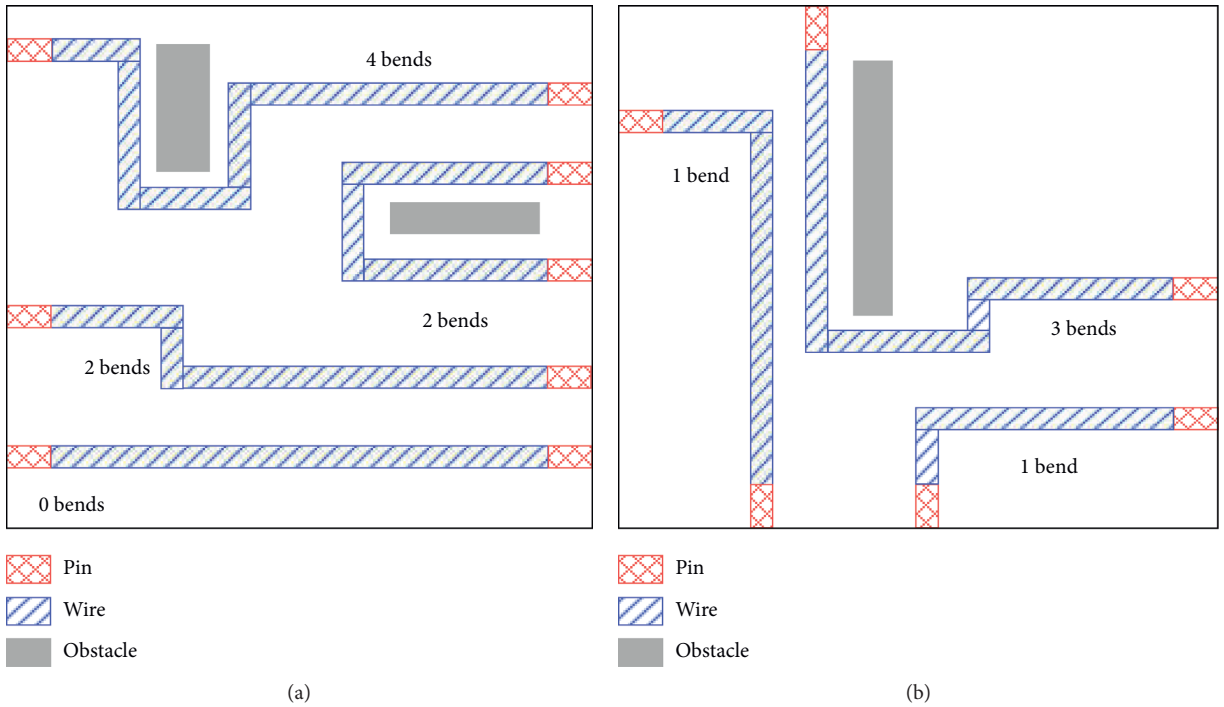

Figure 6: The bits can be categorized into two types based on the preferred directions of pins. (a) The two pins of each bit have the same preferred direction, and the path connecting the two pins has an even number of bends. (b) The two pins of each bit have orthogonally preferred directions, and the path connecting the two pins has an odd number of bends.

decreases dramatically as the demand approaches the capacity but grows slowly in the undercapacity and overcapacity parts.

We determine the order of routing topology according to the number of bends and the weight of path $w(P)$. A smaller number of bends have a higher priority. For each bus, we try the topologies of the bus one by one until the bus is routed successfully. Further, we introduce a new variable x_i for each bit i, where $x_i = \sum_{P \in \mathcal{P}_i^T} x(P)$, and let u' be a copy of u. Then, the global routing problem of the bus with topology T can be formulated as UFP as follows:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n_{\text{bit}}} \sum_{P \in \mathcal{P}_i^T} w(P) \cdot x(P) \\
& x_i - \sum_{P \in \mathcal{P}_i^T} x(P) = 0, \quad 1 \le i \le n_{\text{bit}} \\
\text{s.t.} \quad & \sum_{i=1}^{n_{\text{bit}}} d_i \sum_{P \in \mathcal{P}_i^T:\, e \in P} x(P) \le u'(e) - d(e), \quad e \in E \\
& x_i, x(P) \in \{0,1\}, \quad 1 \le i \le n_{\text{bit}}, P \in \cup_i \mathcal{P}_i^T.
\end{aligned} \tag{2}$$

In the formulated UFP, the objective is to maximize the number of routable bits, and the total weight of all selected paths is as large as possible (i.e., the congestion is as small as possible). The constraints in the first line ensure that at most one path is selected per bit, and the constraints of the second line limit the total demand of bits that can pass through each edge. u' will be increased if all the topologies of the bus fail to be routed. A bus is successfully routed if all bits of the bus are successfully routed (i.e., $x_i = 1, 1 \le i \le n_{\text{bit}}$). Once a bus is successfully routed, we update the demand $d(e)$ and weight $w(e)$ of each edge e and then handle the next bus.

Based on the combinatorial algorithm for UFP in [7], Algorithm 1 provides an algorithm for problem (1). Let $|E| = m$ and $u_{\min}(u_{\max})$ be the minimum (maximum) edge capacity and $d_{\min}, d_{\max}, w_{\min}, w_{\max}$ be the minimum/maximum demand/weight among all bus bits. In Line 1, we first partition the set of bits T into two disjoint sets T_1 and T_2. T_1 consists of bits for which $d_j \le u_{\min}/2$, and the rest of the bits are in T_2. For each bit j and a given path P of bit j, we adopt $F(j,P)$ in [7] to measure the weight gain relative to the added demand load. We set the lower bound α_{lb} and the upper bound α_{ub} on F in Line 3. The order of bits are sorted in Line 6, and then we handle the bits one by one to select a path for each bit in Lines 7–11. $L_{j-1}(e)$ in Line 8 denotes the relative load of edge e after routing bit j.

The time complexity of Algorithm 1 is $O(n_{\text{bit}} \cdot |E_{P^T}|)$, where n_{bit} is the number of bits in a bus and $|E_{P^T}|$ is the number of edges of the paths that have the same routing topology T for the bus. In detail, as can be seen from Algorithm 1, Line 1 requires $O(n_{\text{bit}})$ time, Line 6 requires $O(n_{\text{bit}} \cdot \log(n_{\text{bit}}))$, and Lines 8–9 need $O(|E_{P^T}|)$ time. Besides, since the number of loops in Line 2 and Line 4 is constants, the number of loops in Line 7 is n_{bit}. Hence, Algorithm 1 requires $O(n_{\text{bit}} \cdot |E_{P^T}|)$ time for each bus.

Theorem 1. *Algorithm 1 is an $O(\sqrt{m})$ approximation algorithm for the UFP.*

Proof. Consider an optimal solution routing bits in $\mathcal{Q} \subseteq T$. For each $j \in \mathcal{Q}$, let $\mathcal{Q}_j$ be the route chosen for j in the optimal solution. The total weight of either $\mathcal{Q} \cap T_1$ or $\mathcal{Q} \cap T_2$ is at least $w(\mathcal{Q}/2)$. Denote that set by $\mathcal{Q}'$ and its index by $i' \in \{1,2\}$, and let $\alpha' = 2^{k'}$ be the highest such that $w(\{j \in \mathcal{Q}' | F(j, \mathcal{Q}_j) > \alpha'\}) \ge w(\mathcal{Q})/4$. Let $\mathcal{Q}_{\text{high}}' = \{j \in \mathcal{Q}' \mid F(j, \mathcal{Q}_j) > \alpha'\}$ and $\mathcal{Q}_{\text{low}}' = \{j \in \mathcal{Q}' | F(j, \mathcal{Q}_j) \le 2\alpha'\}$ be sets of higher and lower quality routes in $\mathcal{Q}'$. According to the definition of F, we have $w(\mathcal{Q}_{\text{low}}') \le 2\alpha' \sum_e 1 = 2m\alpha'$, where the inequality is true since an optimal solution cannot overflow an edge. Therefore, we have $w(\mathcal{Q}) \le 8m\alpha'$. In addition, since $F(j, P_j) > \alpha'$ for every $j \in \mathcal{P}$, according to [7], we have $w(\mathcal{P}) = \alpha' \sum_e L_l(e) \ge (1/4)\sqrt{m}\alpha'$. By combining the two inequalities, we get $(w(\mathcal{Q})/w(\mathcal{P})) \le 32\sqrt{m} = O(\sqrt{m})$. □

3.2.3. Rip-Up and Reroute. Rip-up and reroute is a basic routing technique and is usually combined with the negotiation technique. The negotiation-based rip-up and reroute is widely used in global routing [8, 9], track assignment [10], and detailed routing [11] and has been shown to be effective and efficient to improve the routing quality.

At each rip-up and reroute iteration, we first identify and mark a set of buses with overflowed edges or excessive routing cost (equation (9)) that need to be ripped up and rerouted. Rerouting the buses that do not overflow but have excessive routing costs can not only reduce the routing cost but also free up routing resource for other overflowed buses. Then, the marked buses are sorted in decreasing order based on the score defined as follows:

$$\begin{aligned}
S_{\text{order}}(B_i) = {} & C_1 \cdot ne_{\text{of}}(B_i) + \alpha \cdot C_w(B_i) \\
& + \beta \cdot C_s(B_i) + \gamma \cdot C_c(B_i),
\end{aligned} \tag{3}$$

where $\alpha \cdot C_w(B_i) + \beta \cdot C_s(B_i) + \gamma \cdot C_c(B_i)$ is the routing cost defined in equation (9), $ne_{\text{of}}(B_i)$ denotes the number of overflowed edges passed by bus B_i in the previous iteration, and C_1 is a user-defined parameter which is set as $\alpha + \beta + \gamma$.

In addition, the history-based cost function for each routing edge e in [8] is adopted, which is defined as

$$\text{cost}(e) = b(e) + h(e) \times p(e) + \text{vc}(e), \tag{4}$$

where $b(e)$ is the wire length cost, $h(e)$ is the history cost, $p(e)$ is the current penalty cost, $h(e) \times p(e)$ denotes the congestion cost of edge e, and $\text{vc}(e)$ is the via cost. The weight of edge e is set as $w(e) = (1/\text{cost}(e))$, and we reroute a bus by solving problem (2).

We repeat the rip-up and reroute process until there is no overflowed edge or excessive routing cost or the given maximum number of iterations is reached. Since we reroute a bus by solving the UFP (2) and the required runtimes is $O(n_{\text{bit}} \cdot |E_{P^T}|)$, the rip-up and reroute stage requires $O(n_{rb} \cdot n_{\text{bit}} \cdot |E_{P^T}|)$, where n_{rb} is the total number of buses that need to be ripped-up and rerouted.

After obtaining a 2D global routing solution, we extend the layer assignment method in [12] to map the solution from the projected plane to the original multiple layers. Note that, within each segment of a bus, we ensure that the wires of different bits are assigned to the same layer.

3.3. Track Assignment. After obtaining desired routing regions for each bus in the global routing stage, we propose a topology-aware track assignment in this subsection to allocate tracks to each segment of buses under the guidance of the global routing result. In this track assignment stage, we treat the array of all g-cells in a row or column of a routing layer as a panel, and each straight wire that passes through one or more g-cells is regarded as an iroute.

3.3.1. Initial Track Assignment. Since all bits in each bus need to be routed with the same routing topology, we handle the buses one by one to assign the tracks to each segment. For each bus, each segment consists of the set of iroutes of different bits that have the same sequence when traced from source pin to sink pin.

In order to maintain the relative order of the iroutes in each segment, our initial track assignment for each segment is as follows. First, we sort the iroutes of each segment in the same order or in the reverse order of bits. Both orders are tested because the results of each order may be different, and the best results are adopted. Then, based on the sorted order, for each iroute, we collect the valid tracks in the panels and calculate the cost of assigning the iroute to each valid track. A track is valid if the width constraint of the track is greater than or equal to the wire width of the iroute. Finally, we select a valid track with the minimum cost to accommodate the iroute.

The cost function for assigning the iroute ir to the track t is defined as

$$\begin{aligned}\mathrm{cost}(\mathrm{ir},t) = {} & \mathrm{wl}(\mathrm{ir},t) + C_2 \cdot \mathrm{ol}(\mathrm{ir},t) + C_3 \cdot \mathrm{blk}(\mathrm{ir},t) \\ & + C_4 \cdot \mathrm{cp}(\mathrm{ir},t),\end{aligned} \tag{5}$$

where $\mathrm{cost}(\mathrm{ir},t)$ is the total cost of assigning track t to iroute ir, $\mathrm{wl}(\mathrm{ir},t)$ is the wire length cost, $\mathrm{ol}(\mathrm{ir},t)$ is the overlap cost, $\mathrm{blk}(\mathrm{ir},t)$ is the blocked interval cost, $\mathrm{cp}(\mathrm{ir},t)$ is the compactness cost, and C_2, C_3, and C_4 are the user-defined constants which are set as 0.2, 1000, and 1, respectively.

The definition of wire length cost is adopted from the work [10]. Because the routing tracks may be nonuniform or even overlapping, the overlap cost of an iroute being assigned to a track is modified from the work [10], which is determined not only by the overlapping iroutes on that track but also by the overlapping iroutes on other tracks. In addition, since some tracks may only cover a partial design, the blocked interval cost is the sum of blockage cost defined in [10] and the length of the iroute ir that is not on the tracks. According to the 2018 CAD Contest at ICCAD [5], the wires of all bits in a bus should be as compact as possible. Thus, we define the compactness cost to make each bus more compact and reserve more free space for other buses.

Figure 7 illustrates the calculation of the compactness cost. We assume without loss of generality that the panels are horizontal. For the first and last segments of each bus, since the iroutes eventually need to be connected to the corresponding pins in the detailed routing stage, the compactness cost is set as the vertical distance between the iroute and the corresponding pin. For example, Figure 7(a) shows the first segment of a bus, where pin p_1 and iroute a_1 belong to the first bit and pin p_2 and iroute a_2 belong to the second bit. The compactness costs of iroute a_1 on tracks T_1, T_2, and T_3 are 0, d_1, and $d_1 + d_2$, respectively, and the compactness costs of iroute a_2 on tracks T_1, T_2, and T_3 are $d_1 + d_2$, d_2, and 0, respectively.

For the rest of the segments in the bus that do not need to connect pins, the compactness costs of the iroutes are related to the order in which the iroutes of bits are assigned. If we handle iroute a_1 before iroute a_2 in Figure 7(b), the compactness cost of each iroute is the vertical distance between the iroute and the upper boundary of the panel. Conversely, if we handle a_2 before a_1, then the compactness cost of each iroute is the vertical distance between the iroute and the lower boundary of the panel. In addition, if the iroutes of a segment are distributed in multiple panels as shown in Figure 7(c), the compactness cost of each iroute in the bottommost panel is the vertical distance between the iroute and the upper boundary of the panel, the compactness cost of each iroute in the topmost panel is the vertical distance between the iroute and the lower boundary of the panel, and the compactness cost of each iroute in the rest panels is 0.

3.3.2. Rip-Up and Reassignment. After the initial track assignment, there may be overlaps between the iroutes. Therefore, we extend the negotiation-based track assignment in the work [10] to minimize the overlaps and wire length while keeping the relative order of the iroutes in each segment.

The cost function for reassignment is defined as

$$\begin{aligned}\mathrm{cost}_{\mathrm{his}}(\mathrm{ir},t) = {} & \theta_{\mathrm{wl}} \cdot \mathrm{wl}(\mathrm{ir},t) + \theta_{\mathrm{ol}} \cdot \mathrm{ol}(\mathrm{ir},t) \\ & + \theta_{\mathrm{blk}} \cdot \mathrm{blk}(\mathrm{ir},t) + \theta_{\mathrm{cp}} \cdot \mathrm{cp}(\mathrm{ir},t) + \theta_{\mathrm{his}} \cdot \mathrm{his}(\mathrm{ir},t),\end{aligned} \tag{6}$$

where $\mathrm{wl}(\mathrm{ir},t)$, $\mathrm{ol}(\mathrm{ir},t)$, $\mathrm{blk}(\mathrm{ir},t)$, and $\mathrm{cp}(\mathrm{ir},t)$ are the same as the definition in equation (5), and $\mathrm{his}(\mathrm{ir},t)$ is the history cost from the work [10]. The user-defined parameters $\theta_{\mathrm{wl}}, \theta_{\mathrm{ol}}, \theta_{\mathrm{blk}}, \theta_{\mathrm{cp}}$, and θ_{his} are used to balance the cost components. Both θ_{wl} and θ_{cp} are initialized as 1 and gradually decreased to 0.1 as the iteration increases, and θ_{ol} is initialized as 0.1 and gradually increased to 1 as the iteration increases. Besides, θ_{blk} is a very large constant and θ_{his} is set as 1. Through the control of these parameters, we can reduce the overlaps with less wire length and compactness cost at early iterations and focus more on overlap reduction at late iterations.

In order to maintain the same topology for all bus bits, we always keep the relative order of the iroutes in each segment during the rip-up and reassignment stage. However, reassigning an iroute while keeping the relative order of iroutes in the segment may fall into a local optimum, due to the fact that the solution space is limited and we may always select the same set of iroutes or always try to assign an iroute to a small set of tracks. Therefore, in each iteration, we may rip up and reassign multiple iroutes to reduce the probability of falling into local optimum. Specifically, when an iroute has no other tracks that can be assigned to or these tracks have been tried several times by this iroute, we will also rip up and reassign some of the iroutes that adjacent to this iroute. The time complexity of reassigning an iroute is $O(n_{pt})$ at each iteration, where n_{pt} is the number of tracks in the panel that the iroute is located. Since we may rip up and reassign multiple iroutes simultaneously to reduce the probability of falling into local optimum, if we handle k iroutes simultaneously, the time complexity is $O(n_{pt}^k)$. However, since n_{pt} is not too large (tens to hundreds) and we set k as 3, the running time of this stage depends mainly on the number of iroutes that need to be reassigned.

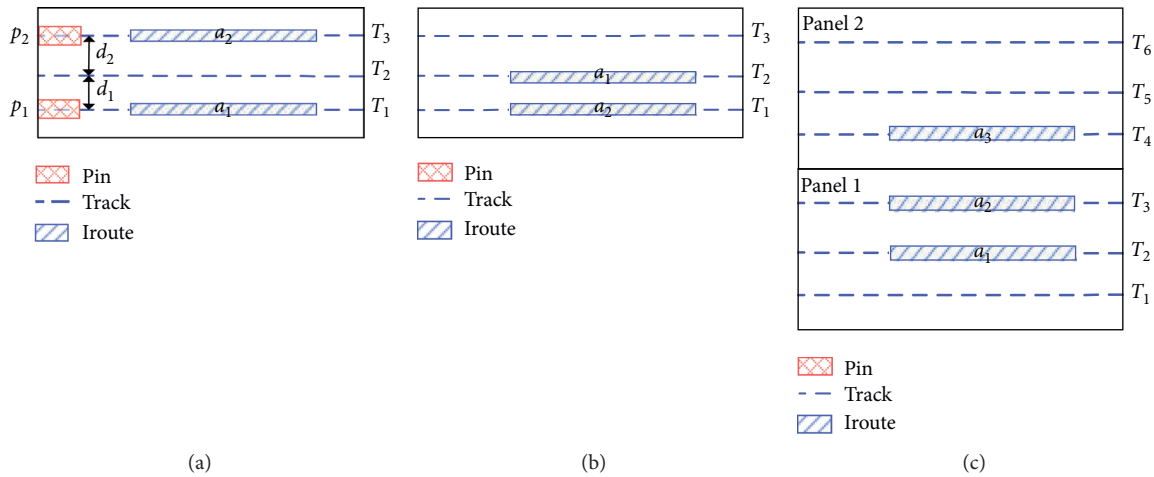

FIGURE 7: Three cases of compactness cost calculation. (a) Iroutes of the first or last segment, which eventually needs to be connected to the corresponding pins in the detailed routing stage. (b) All iroutes of a segment are in the same panel. (c) The iroutes of a segment are distributed in multiple panels.

For example, assume that iroute a_2 in Figure 7(c) needs to be reassigned. Since the relative order of the iroutes of different bits in a segment should be maintained, iroute a_2 can only be placed between iroute a_1 and iroute a_3. That is, if iroute a_1 is not reassigned to another track, then iroute a_2 cannot be reassigned. Therefore, we rip up both the iroutes a_1 and a_2, and then iroutes a_1 and a_2 can be reassigned to the tracks T_1 and T_2, respectively.

3.4. Detailed Routing. After track assignment, we need to connect the components of each bit to obtain the final routing result. A bit component is a pin or an iroute of the bit. Algorithm 2 gives the framework of our detailed routing. In Line 3, in order to preserve the same routing topology for all bus bits and honor the global routing result, the components of each bit are sorted according to the trace order of global routing paths from the source pin to the sink pin, and then we only need to connect the adjacent components of each bit one by one.

In Line 4, we adopt L-shaped [13], Z-shaped [13], and the 3-bend routing [14] to connect the adjacent components, because it is easy to control the same topology for all bus bits and is very efficient by using these predefined pattern routing. After that, we use two-stage negotiation-based rip-up and reroute to iteratively improve the solution quality in Lines 8–16. ite1 and ite2 in Line 8 indicate the maximum number of iterations for the first stage and the second stage of rip-up and reroute, respectively. In the first stage, we rip up every two adjacent components that have the overlapping wires and reroute them through the patterns routing while maintaining the same routing topology. After each iteration, we increase the history cost of the overlapped interval on a track according to the number of overlapped wires. As a result, a track with a higher history cost tends to have less chance to be routed, and the bits with alternative routes are forced to use other tracks. In the end, the bit that most needs to use this track will eventually use it.

Since the limited search space of the patterns routing and only allowing to rip up and reroute the adjacent components in the first stage rip-up and reroute, it is possible to reduce the wire overlap if some restrictions are removed. In the second stage, we use the A^* algorithm [15] to search for the paths and allow paths to be outside the global routing guide. In addition, we also allow some iroutes extracted in the track assignment stage to be removed, thus increasing the freedom of routing. For example, if we remove the second component of a bit, then we are required to find a path to connect the first component and the third component. When a part of a bit is rerouted, we check whether other bits in the bus are the same as its topology. If not, we will adjust the routing paths of other bits based on the path of that bit. We repeat the rip-up and reroute process until all the buses are routed successfully or the given maximum number of iterations is reached.

Finally, we construct a conflict graph in which each bus is regarded as a vertex, and each edge represents the conflict between two buses. We iteratively rip up the bus (vertex) with the largest degree and its associated edges until there are no conflict edges in the graph, and the final routing result is obtained.

4. Experimental Results

To evaluate our proposed bus routing algorithm, we implemented our algorithm in the C++ programming language and tested it on the benchmarks (including the hidden cases) of the 2018 CAD Contest at ICCAD on Obstacle-Aware On-Track Bus Routing [5]. Table 1 lists the benchmark statistics, where "#Layer," "#Track," "#Obstacle," "#Bus," "#Bit," and "#Pin" give the total numbers of layers, tracks, obstacles, buses, bits, and pins, respectively.

The score function in the contest [5] is adopted to evaluate the quality of bus routing results, which consists of routing cost C_r, spacing violation penalty P_s, and fail routing penalty P_f. That is,

Input: The graph $G(V, E, u)$, all bits of a bus.
Output: The set P_{best} of paths that connecting the bits.
(1) Partition the set of bits T into two disjoint sets T_1 and T_2;
(2) **for** $i = 1, 2$ **do**
(3) $\alpha_{\text{lb}} \leftarrow w_{\min}/|V|, \alpha_{\text{ub}} \leftarrow w_{\max} u_{\max}/d_{\min}$;
(4) **for** each k from $\lfloor \log \alpha_{\text{lb}} \rfloor$ to $\lceil \log \alpha_{\text{ub}} \rceil$ **do**
(5) $\alpha \leftarrow 2^k$;
(6) Sort the bits in T_i according to a nonincreasing order of w_j/d_j;
(7) **for** each $j \in T_i$ **do**
(8) **if** $\exists$ path P of bit j s.t. $F(j, P) > \alpha$ and $\forall e \in P, L_{j-1}(e) + d_j/u(e) \le 1$ **then**
(9) Route the bit on P and for $e \in P$ set $L_j(e) = L_{j-1}(e) + d_j/u(e)$;
(10) Update P_{best};
(11) **end if**
(12) **end for**
(13) **end for**
(14) **end for**

Algorithm 1: Unsplittable flow problem solving.

Input: A set of components of all bits in all buses.
Output: Final routing result.
(1) **for** each bus B_i **do**
(2) **for** each bit b_{ij} **do**
(3) Sort the components;
(4) Connect the adjacent components with the same topology;
(5) **end for**
(6) **end for**
(7) the components that wires overlap, $i \leftarrow 0$;
(8) **while** $O_c \neq \varnothing$ and $i <$ ite1 + ite2 **do**
(9) **if** $i <$ ite1 **then**
(10) Reroute1(O_c);
(11) **else**
(12) Reroute2(O_c);
(13) **end if**
(14) Update history cost andO_c;
(15) $i \leftarrow i + 1$;
(16) **end while**

Algorithm 2: Detailed routing framework.

Table 1: Statistics of the 2018 CAD Contest at ICCAD benchmarks.

Benchmark	#Layer	#Track	#Obstacle	#Bus	#Bit	#Pin
beta_1	3	49209	159	34	1260	2520
beta_2	3	49209	0	26	1262	2524
beta_3	3	22732	555108	60	665	1330
beta_4	3	22732	0	62	698	1396
beta_5	4	54150	0	6	1964	3928
final_1	3	81226	0	18	1032	2064
final_2	3	14209	0	70	1285	2570
final_3	4	21379	0	47	852	1704

Table 2: Experimental results.

Benchmark	First place				Second place				Third place				Ours				
	C_r	P_s	P_f	S	C_r	P_s	P_f	S	C_r	P_s	P_f	S	C_r	P_s	P_f	S	CPU (s)
beta_1	689	280	0	969	701	5096	0	5797	641	8744	4000	13385	812	432	0	1244	10
beta_2	515	760	0	1275	563	4904	0	5467	484	9472	2000	11956	626	224	0	850	8
beta_3	1936	0	0	1936	2024	0	0	2024	1999	1928	0	3927	1905	0	0	1905	3600
beta_4	2192	0	0	2192	2271	0	0	2271	2250	1048	0	3298	2376	184	0	2560	3600
beta_5	119	1848	0	1967	95	616	2000	2711	98	1216	2000	3314	95	0	2000	2095	45
final_1	327	830	2000	3157	367	2750	2000	5117	252	0	10000	10252	341	430	2000	2771	3600
final_2	1824	4500	8000	14324	1890	2990	8000	12880	1976	6910	0	8886	2076	1470	6000	9546	2521
final_3	2966	490	10000	13456	2678	300	2000	4978	4238	20	24000	28258	2674	540	10000	13214	3600
Normalized	1.09				2.24				4.57				1.00				

$$S = C_r + P_s + P_f. \tag{7}$$

The three parts of the score function are calculated as follows:

$$\begin{cases} C_r = \sum_{B_i \in \mathcal{B}} \left(\alpha \cdot C_w(B_i) + \beta \cdot C_s(B_i) + \gamma \cdot C_c(B_i)\right), \\ P_s = \text{number of spacing violations} \times \delta, \\ P_f = \text{number of route fail buses} \times \varepsilon, \end{cases} \tag{8}$$

where $\alpha, \beta, \gamma, \delta, \varepsilon$ are five weighting parameters given in the input data, and the values of these parameters may vary from different benchmarks. For each bus B_i, the wire length cost $C_w(B_i)$, the segment cost $C_s(B_i)$, and the compactness cost $C_c(B_i)$ are defined as follows:

$$\begin{cases} C_w(B_i) = \dfrac{\sum_j^{\text{All bits of bus } B_i} (\text{wire length of bit } j/\text{half parameter wire length of bit } j)}{\#\text{bits of bus } B_i}, \\ C_s(B_i) = \dfrac{\#\text{segment of bus } B_i}{\text{lower bound of } \#\text{segment of bus } B_i}, \\ C_c(B_i) = \dfrac{\sum_j^{\text{All bits of bus } B_i} (\text{width of segment } j/\text{lower bound width of segment } j)}{\#\text{segment of bus } B_i}. \end{cases} \tag{9}$$

Ideally, if a bus B_i is routed with the minimum wire length ($C_w(B_i) = 1$) and the minimum number of segments ($C_s(B_i) = 1$) and all segments are routed with widths close to the lower bound ($C_c(B_i) = 1$), then the routing cost C_r of the perfectly routed bus is close to $\alpha + \beta + \gamma$.

The experimental results of the top 3 teams of the 2018 CAD Contest at ICCAD [5] and ours are listed in Table 2. Our algorithm was run on a Linux workstation with a 2.40 GHz Intel Xeon CPU and 64 GB memory, and the results of top 3 teams of the contest are provided by the contest organizer. Since the binaries of top 3 teams are not available for us, we do not report their runtime. Nevertheless, the specified time for each test case is one hour according to the contest [5], and our program will be killed if the runtime exceeds the specified time.

In Table 2, the columns "First place," "Second place," "Third place," and "Ours" give the corresponding routing results generated by the first place, second place, and third place of the contest [5] and our algorithm, respectively. The latest evaluation script (eval_1.0-a8) provided by the contest [5] was used to obtain the routing cost "C_r," spacing violation penalty "P_s," fail routing penalty "P_f," and score "S." It can be seen from Table 2 that all the buses are successfully routed by our algorithm for the tested cases beta_1, beta_2, beta_3, and beta_4. Particularly, on average, our algorithm outperforms the top 3 teams by 9%, 124%, and 357% in the final scores, respectively. The experimental results show that our proposed bus routing algorithm is effective.

5. Conclusions

In this paper, we have presented an effective algorithm to solve the topology-aware bus routing problem considering the existence of both nonuniform track configuration and

obstacles. We first presented a track handling technique to unify the nonuniform routing track configuration together with obstacles. Then, we have formulated the topology-aware routing problem of single bus as UFP, which is integrated into a negotiation-based global routing problem of determining the desired routing regions for each bus. Moreover, we presented a topology-aware track assignment method which can allocate the tracks to each segment of buses regarding the guidance of the global routing result. Lastly, a detailed routing scheme has been presented to connect the segments of each bus. We have evaluated our routing results with ICCAD benchmark suites. Compared with the state-of-the-art methods, the experimental results have shown that our proposed method achieves the best overall score within the specified time.

Data Availability

The data used in this study can be accessed via http://iccad-contest.org/2018/problems.html.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities of China under Grant 2242021k30031, National Key Research and Development Project of China under Grant 2018YFB22022704, National Science Foundation of China (nos. 61977017 and 61772005), and Outstanding Youth Innovation Team Project for Universities of Shandong Province under Grant 2020KJN008.

References

[1] G. Georgiev, A. Chatterjee, and G. Iannacchione, "Exponential self-organization and moores law: measures and mechanisms," *Complexity*, vol. 2017, Article ID 8170632, 9 pages, 2017.

[2] Y. Tian and T. Watanabe, "Improved delay-matching bus routing by using multi-layers," in *Proceedings of the International Conference on Electronics Packaging and iMAPS All Asia Conference (ICEP-IAAC)*, pp. 708–713, Kyoto, Japan, April 2015.

[3] T. Yan and M. D. F. Wong, "Bsg-route: a length-matching router for general topology," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 499–505, San Jose, CA, USA, November 2008.

[4] R. Zhang, T. Pan, L. Zhu, and T. Watanabe, "A length matching routing method for disordered pins in pcb design," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, Chiba, Japan, January 2015.

[5] A. Liao, H.-Y. Chang, O. Chi, and J. Wang, ICCAD 2018 CAD contest: obstacle-aware on-track bus routing, http://iccad-contest.org/2018/problems.html, 2018.

[6] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*, Auerbach Publications, Boca Raton, FL, USA, 2008.

[7] Y. Azar and O. Regev, "Combinatorial algorithms for the unsplittable flow problem," *Algorithmica*, vol. 44, no. 1, pp. 49–66, 2006.

[8] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-route 2.0: a robust global router for modern designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1931–1944, 2010.

[9] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 5, pp. 709–722, 2013.

[10] M.-P. Wong, W.-H. Liu, and T.-C. Wang, "Negotiation-based track assignment considering local nets," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, Macao, China, January 2016.

[11] F.-K. Sun, H. Chen, C.-Y. Chen, C.-H. Hsu, and Y.-W. Chang, "A multithreaded initial detailed routing algorithm considering global routing guides," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 81: 1–81:7, San Diego, CA, USA, November 2018.

[12] 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.

[13] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 7, pp. 777–790, 2002.

[14] Y. Xu, Y. Zhang, and C. Chu, "Fastroute 4.0: global router with efficient via minimization," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference*, Yokohama, Japan, January 2009.

[15] A. Hetzel, "A sequential detailed router for huge grid graphs," in *Proceedings of the Conference on Design, Automation and Test in Europe*, Paris, France, February 1998.