MiniDeviation: An Efficient Multi-Stage Bus-Aware Global Router

Weida Zhu*, Xinghai Zhang*, Genggeng Liu*, Wenzhong Guo* and Ting-Chi Wang†
*College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
†Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
Email: {zhu_weida, zhang_xinghai, liu_genggeng}@126.com, guowenzhong@fzu.edu.cn, tcwang@cs.nthu.edu.tw

***Abstract*—As the number of signal nets increases significantly, global routing of buses becomes an increasingly important and difficult problem. In this paper, to match the timing of buses, we propose an efficient multi-stage bus-aware global routing algorithm called *MiniDeviation* that is based on several techniques: 1) a deviation-driven segment shifting, 2) a multi-stage double maze routing strategy, and 3) a post-routing scheme. Compared with the existing algorithms, the experimental results show that the proposed global router achieves the best results for both total wirelength deviation and total overflow.**

I. INTRODUCTION

The constantly evolving modern VLSI technology continues to push the complexity of integrated circuits to new heights and brings new challenges for on-chip interconnection. Furthermore, with the increasing trend of Intellectual Property integration, the number of buses among different modules on a chip also grows rapidly, making global routing of buses a major task of importance to timing in modern circuit design. Recently, many global routers have been proposed in the literature [1]-[4]. However, these global routing algorithms, without taking the bus into consideration, may cause the signal propagation timing deviation to be serious on the global routing problem. Therefore, it is extremely required to develop a better global router that considers buses to minimize the wirelength deviation.

For bus routing, some research has been studied. Mo and Brayton [5] proposed a bus routing algorithm to re-use routing topology of the different bits in a bus. Yan and Wong [6] presented BSG-Route that uses a placement structure and bounded-slice line grid to solve the mathematical problem. For obstacle-aware bus routing problem, a heuristics algorithm that adopts longest common subsequence and commodity flow was presented by Zhang et al. [7]. In addition, to stimulate the research and development of bus routing techniques for modern designs, ICCAD announced a topology-matching bus routing problem in 2018. Hsu et al. [8] presented a bus routing algorithm based on directed acyclic graph to connect a bus in the specific topology. Chen et al. [9] proposed a maze routing named MARCH by routing all the bits of a bus concurrently for topology consistency.

However, these routing algorithms [1]-[9] mentioned above do not consider the length-matching of buses in global routing stage. Although a bus-aware global router (BGR) was proposed to handle the length-matching issue by modifying NTHU-Route 2.0 [3], it generated excessive wirelength deviation and more overflow owing to not effectively considering wirelength deviation [10]. In addition, total overflow reduction is an important goal. Therefore, this paper focuses on the global routing problem for minimizing total wirelength deviation and total overflow. The major contributions of this paper are as follows.

1). We propose *MiniDeviation*, which processes a bus bit by bit, to minimize total wirelength deviation and total overflow.
2). A deviation-driven segment shifting is designed to reduce the congestion considering wirelength deviation.
3). A multi-stage double maze routing strategy is applied to eliminate overflow and control wirelength deviation during rip-up/rerouting stage.
4). We present an effective post-routing scheme to further improve the routing solution quality.

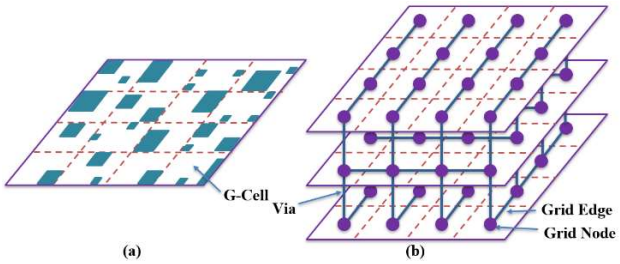

Fig. 1: G-Cells and corresponding 3D global routing grid graph.

The experimental results show that MiniDeviation achieves the best results. Compared with NTHU-Route 2.0 [3] and BGR [10], MiniDeviation achieves 57.86% and 31.30% less total overflow for the benchmarks with overflow, respectively. In terms of total wirelength deviation, MiniDeviation is 4.05 and 1.61 times better than NTHU-Route 2.0 and BGR on average, respectively.

The remainder of this paper is organized as follows. The problem formulation of this work is introduced in Section II. In Section III, we present an efficient multi-stage bus-aware global router that adopts a deviation-driven segment shifting, a multi-stage double maze routing strategy, and an effective post-routing scheme. Experimental results are discussed in Section IV, and Section V concludes this paper.

II. PROBLEM FORMULATION

In global routing, the routing region is partitioned into a set of rectangular grid cells called G-Cells (see Fig. 1(a)). Therefore, a grid graph is defined as $G = (V, E)$, where the nodes $v \in V$ represent the routing grid cells and the edges $e \in E$ represent connections of adjacent grid cell pairs. Fig. 1(b) shows a three-layer grid graph of a circuit containing 4×4 G-Cells in each metal layer. In the graph, each metal layer is dedicated to either horizontal or vertical wires. Besides, any two adjacent layers are connected by vias.

The capacity of a grid edge, $c(e)$, is defined as the maximum number of wires that can cross the grid edge e. The demand of e, $d(e)$, is defined as the actual number of wires crossing the grid edge e. An overflow occurs when $d(e)$ exceeds $c(e)$. The overflow of e, $o(e)$, is defined as follows:

$$o(e) = \begin{cases} d(e) - c(e), if\ d(e) > c(e) \\ 0 \quad\quad , otherwise \end{cases} \tag{1}$$

For a bus B_i carrying r signal bits and including a set PG^i of q pin groups, it has one source pin group (PG^i_0) and q-1 sink pin groups (PG^i_1, PG^i_2, …, PG^i_{q-1}). Each PG^i_j is a set of aligned r pins belonging to the bus B_i. To make it explicit, each bus bit is defined as a bus net. Besides, the number of pins in each $PG^i_j \in PG^i$ is identical and equal to the number of bits in bus B_i. In order to meet the timing consistency, it is necessary to transfer each signal bit of PG^i_0 to PG^i_j ($1 \le j \le q$-1), as concurrently as possible. That is to say, the wirelength of each bus pin pair (BPP^i_j) between PG^i_0 and PG^i_j is to be as similar as possible. The consistency of the timing can indicate the merits of the routing solutions, and the wirelength deviation of bus B_i is defined as follows:

$$WD(B_i) = \sum_{j=1}^{q-1} \sum_{k=1}^{r} (MWPG^i{}_j - WPG^i{}_j\langle k \rangle) \tag{2}$$

978-1-7281-6083-2/20/$31.00 ©2020 IEEE

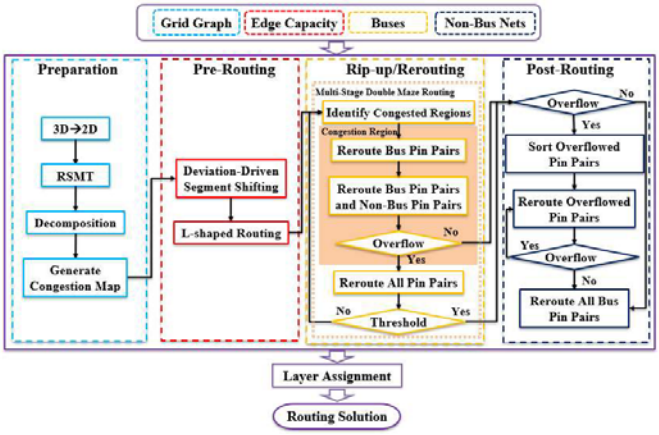

Fig. 2: Flow of MiniDeviation.

where $WPG^i_j<k>$ is the wirelength of the k^{th} bus pin pair in PG^i_0 and PG^i_j of bus B_i, and $MWPG^i_j$ is the maximum wirelength of all bus pin pairs between PG^i_0 and PG^i_j of bus B_i.

According to the routing grid cell where a pin is located, all pins map to the corresponding nodes of the grid graph $G = (V, E)$. Global routing is to find paths on $G = (V, E)$ that connect the pins inside the G-Cells for all nets. Few overflow and short wirelength deviation imply high routability and bus time consistency. Thus, the major objectives of bus-aware global routing are to minimize the total overflow for every grid edge and total wirelength deviation for every bus. In addition, total wirelength of all nets and runtime are other important metrics for global routing. We formally define the bus-aware global routing problem below.

The Bus-Aware Global Routing Problem: A grid graph $G = (V, E)$, the capacity $c(e)$ of each grid edge e, a set $NB = \{NB_1, NB_2, \ldots, NB_m\}$ of m non-bus nets and a set $B = \{B_1, B_2, \ldots, B_n\}$ of n buses are given. A bus-aware global router focuses on generating a routing solution with minimizing the total overflow (TOF) and total wirelength deviation (TWD). That is, the quality of a bus-aware global routing solution is evaluated by the following quantities:

$$TOF = \sum_{e \in E} o(e) \tag{3}$$

$$TWD = \sum_{i=1}^{n} WD(B_i) \tag{4}$$

III. ALGORITHM

Our router MiniDeviation converts the 3D grid graph into the 2D grid graph, solves the 2D routing problem, and then assigns the 2D solution to the 3D solution for each net by a layer assignment method which is the same as the one in [3]. The flow of our global router is shown in Fig. 2 and is divided into five stages: (1) preparation, (2) pre-routing, (3) rip-up/rerouting, (4) post-routing, (5) layer assignment. The first four stages are introduced in the following subsections.

A. Preparation

Since it is time-consuming to directly solve the 3D routing problem, firstly, the common 3D-to-2D capacity projecting approach is adopted in the preparation stage. Secondly, for each net, we generate its rectilinear Steiner minimal tree by FLUTE [11] and then decompose all Steiner trees into pin pairs. Finally, a congestion map is constructed in the following manner, which is used during the whole procedure. If the two pins of a pin pair are connected by a straight path, we assign the demand value 1 to all edges passed by the path on the grid graph. Otherwise, we assign the demand value 0.5 to all edges along the bounding box of the pin pair. The main task of the preparation stage is to generate the congestion map and a set of pin pairs.

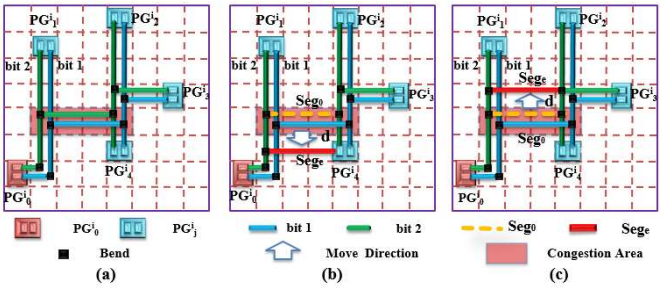

Fig. 3: Deviation-driven segment shifting.

B. Pre-Routing

After generating the congestion map, in the pre-routing stage, in order to get a better routing topology for each pin pair, a deviation-driven segment shifting is proposed to reduce the congestion with considering wirelength deviation by changing the topology of each net, which is different from the method proposed in [1].

The idea of segment shifting is sliding some segments out of the congested regions without increasing the Steiner tree wirelength according to the congestion map. Although the segment shifting can reduce congestion, different from a non-bus net, moving a segment of a bus bit for a bus causes the change of wirelength deviation. To reduce the calculation complexity, it is worth noting that the wirelength deviation here only refers to the comparison with the original routing topology of a bus bit, not the exact total wirelength deviation of the bus. As shown in Fig. 3(a), for the horizontal segment of *bit* 2, there are two cases to consider. The first case is that we move Seg_0 of *bit* 2 close to PG^i_0 by distance d and get Seg_e, as shown in Fig. 3(b). Let $YSeg_0/YSeg_e/YPG^i_j$ denotes the y coordinate of $Seg_0/Seg_e/PG^i_j$. There are three types of PG^i_j. (1) If PG^i_j is below Seg_e, the wirelength between PG^i_0 and PG^i_j is reduced by $2d$. However, the wirelength deviation, $d(PG^i_j)$, is increased by $2d$. (2) If PG^i_j is between Seg_0 and Seg_e, the wirelength is increased $2|YSeg_0 - YPG^i_j|$, but $d(PG^i_j)$ is increased by $2|YSeg_0 - YPG^i_j|$. (3) If PG^i_j is above Seg_0, the wirelength and $d(PG^i_j)$ are both unchanged. Then we discuss the second case. We move Seg_0 of *bit* 2 away from PG^i_0 by distance d in Fig. 3(c). (1) If PG^i_j is below Seg_0, the wirelength between PG^i_0 and PG^i_j is increased by $2d$, but $d(PG^i_j)$ is increased by $2d$. (2) If PG^i_j is between Seg_0 and Seg_e, $d(PG^i_j)$ is increased by $2|YSeg_e - YPG^i_j|$. (3) If PG^i_j is above Seg_e, $d(PG^i_j)$ is unchanged. Therefore, $d(PG^i_j)$ is defined below:

$$d(PG^i_j) = \begin{cases} 2d & , YPG^i_j \le \min(YSeg_0, YSeg_e) \\ 0 & , YPG^i_j \ge \max(YSeg_0, YSeg_e) \\ 2\left|\max(YSeg_0, YSeg_e) - YPG^i_j\right| & , otherwise \end{cases} \tag{5}$$

where Seg_0 is the segment before shifting and Seg_e is the segment after shifting.

If distance d is zero, it may cause an overflow, which requires more time to rip-up and reroute and longer wirelength. Aware of this problem, we want to find a trade-off between congestion and wirelength deviation so that the routing quality can be better. Therefore, we need to decide the best position for a segment. For every possible position, we evaluate the total cost of a path and wirelength deviation due to segment shifting. The best position is the segment with the minimal cost. The cost function is set as follows:

$$P_{cost} = \sum_{e \in p} b_e + \sum_{PG^i_j \in PG^i} d(PG^i_j) \tag{6}$$

where b_e is the cost of edge e, p is the routing path of a pin pair, and $d(PG^i_j)$ is the wirelength deviation cost. b_e is defined below:

$$b_e = 1 + \frac{h}{1 + e^{-k \times (d(e) - c(e))}} \tag{7}$$

where h and k are user-defined parameters.

After deviation-driven segment shifting, to get an initial routing solution quickly, an L-shaped routing method [12] is adopted to route pin pairs regardless of whether a routing path causes overflow and wirelength deviation or not.

C. Rip-up/Rerouting

The main task of the rip-up/rerouting stage is to find an overflow-free path for each pin pair. Since L-shaped routing only considers two paths, a pin pair may fail to find good routing paths to avoid the congestion in many cases. Thus, a multi-stage double maze routing strategy is adopted to eliminate total overflow and control wirelength deviation, which is based on resource adjustment of congested regions and resource adjustment of the entire routing area. Hybrid unilateral monotonic (HUM) routing [13] and adaptive multi-source multi-sink mazing routing (AMMMR) [3] are two very efficient techniques to improve the routing solution. Therefore, MiniDeviation adopts HUM to control the increase in wirelength deviation and AMMMR to approach the behavior of maze routing to find paths avoiding overflow.

Algorithm 1 shows the pseudo code of the multi-stage double maze routing strategy. Firstly, we identify congested regions according to the congestion of edges in line 2, which is to determine the set and the order of pin pairs for rip-up and reroute. The method of identifying the congested regions (*CR*) is similar to the way proposed in [3]. Secondly, lines 3 to 14 are to traverse each congested region, finding and rerouting pin pairs with overflow located in congested regions. Lines 6 to 8 are to rip up and reroute the bus pin pairs (*BPP*) with overflow using HUM. If HUM can find a path without overflow, the new path of HUM is accepted; otherwise, the original path is not replaced so that the increase in wirelength deviation is limited. The purpose of this step is to make buses take priority to use those routing resources that are released in the congested regions. And then, for pin pairs in congested regions, HUM and AMMMR are both employed to find an overflow-free path (lines 9 to 13). The purpose is to adjust the congestion in the congested regions. Finally, in order to further coordinate routing resources in the grid graph, HUM is applied for bus pin pairs to find paths with smaller wirelength deviation after releasing routing resources and AMMMR is used for non-bus pin pairs (*NBPP*) to further reduce the overflow in lines 15 to 20.

The cost function plays an important role in averaging congestion and reducing overflow. A good cost function can effectively improve the routing solution. The following history based cost function is adopted in this stage:

$$P_{cost} = \sum_{e \in p} \left(b_e + h_e \times p_e \right) + d_c + n \times v_c \tag{8}$$

where b_e is the base cost of edge e. h_e is a historical cost and increases as e is overflowed. p_e is a penalty term, which amplifies h_e. p is the path of the pin pair. d_c is the wirelength deviation cost. v_c is the via cost. n is the number of bends of p. b_e and v_c are defined as follows:

$$b_e = 1 - e^{-\alpha \times e^{-\beta \times i}} \tag{9}$$

$$v_c = \lfloor 4 \times b_e \rfloor \times v_g \tag{10}$$

where i is the iteration count, α and β are set to 10, 0.05, respectively. v_g is the expected amount of vias and b_e is the base cost of edge e.

The b_e and v_c are adaptive cost functions and decrease as the iteration count increases, which weakens the effect of wirelength and via in turn. Hence the paths with less overflow are encouraged to obtain rather than the paths with shorter wirelength and fewer vias. However, to some extent, since the wirelength reflects the wirelength

Algorithm 1 Multi-Stage Double Maze Routing Strategy

INPUT: a set *PP* of pin pairs

1. **While (** ***NoOverflow*****() ==** *false* **&&** ***MetThreshold*****() ==** *false* **) do**
2. Identify congested regions and sort them to get *CR*;
3. **for (** $k \leftarrow$ 1 to $|CR|$ **) do**
4. $BPP \leftarrow$ set of bus pin pairs $\subseteq CR_k$;
5. $NBPP \leftarrow$ set of non-bus pin pairs $\subseteq CR_k$;
6. **for (** $i \leftarrow$ 1 to $|BPP|$ **) do**
7. $NewPath \leftarrow$ ***HUM*****(**BPP_i**)**;
8. **end for**
9. **for (** $i \leftarrow$ 1 to $|BPP \cup NBPP|$ **) do**
10. $NewPath \leftarrow$ ***HUM*****(**$BPP_i/NBPP_i$**)**;
11. **if (** *NewPath* of $NBPP_i$ has overflow **) then**
12. $NewPath \leftarrow$ ***AMMMR*****(**$NBPP_i$**)**;
13. **end for**
14. **end for**
15. **for (** $i \leftarrow$ 1 to $|PP|$ **) do**
16. **if (** PP_i is bus pin pair **) then**
17. $NewPath \leftarrow$ ***HUM*****(**PP_i**)**;
18. **else**
19. $NewPath \leftarrow$ ***AMMMR*****(**PP_i**)**;
20. **end for**
21. **end while**

deviation, the attenuation of the wirelength factor causes an increase in the wirelength deviation. Thus, it is necessary to add wirelength deviation term to the cost function to control wirelength deviation. d_c is defined as follows:

$$d_c = \omega \times \eta^{(MWPG^i{}_j - WPG^i{}_j < k >)} \tag{11}$$

where ω and η are user-defined parameters, which are set to 0.1 and 1.002, respectively.

The rip-up/rerouting stage does not end until the total overflow is zero or the iteration count reaches the threshold.

D. Post-Routing

In the post-routing stage, there are two possible scenarios. The first case is that the rip-up/rerouting stage fails to solve all the overflow, and then we use the maze routing to rip-up and reroute the pin pairs with overflow. Otherwise, we only focus on finding a minimum wirelength deviation path for each bus pin pair as much as possible without increasing the number of overflow. Specifically, we rip-up and reroute all buses. Only the new path whose wirelength is equal to half-perimeter of the pin pair can replace the original path.

We adopt the increasing order of the bounding box of pin pair as the net order. Since the wirelength of a pin pair is longer, it is more possible to find an overflow-free or a shorter wirelength path. In this stage, the cost function is set as follows:

$$c(e) = \begin{cases} C, c(e) \le d(e) \\ 0, c(e) > d(e) \end{cases} \tag{12}$$

where C is an extremely large constant, which is set to 1000, thus ensuring that the number of edges with overflow does not increase.

IV. Experimental Results

We implemented the proposed algorithm in C++ on a 2.0 GHz Inter Xeon-based Linux machine with 64 GB memory and tested our program on benchmarks from [10]. Table I shows the characteristics of all benchmarks. (*MinBit/MaxBit*: the minimum/maximum number of bits a bus has. *Bus*: the number of buses. *TotalBit*: the quantity of bits in all buses. *Net*: the quantity of all nets. *G-Cell*: the scale of routing area.)

TABLE I: Characteristics of Benchmarks

Benchmark	MinBit	MaxBit	Bus	TotalBit	Net	G-Cell
adaptec1	8	64	50	1568	219794	324*324
adaptec2	8	64	50	1632	260159	424*424
adaptec3	8	64	50	1512	466295	774*779
adaptec4	8	64	50	1416	515304	774*779
adaptec5	8	64	50	1432	867441	465*468
bigblue3	8	64	50	1696	1122340	555*557
bigblue4	8	64	50	1736	2228903	403*405
newblue2	8	64	50	1712	463213	557*463
newblue4	8	64	50	1208	636195	455*458
newblue5	8	64	50	1384	15257553	637*640
newblue6	8	64	50	1472	1286452	463*464

TABLE II: Total Overflow Comparison

Benchmark	Total Overflow			Improvement	
	NTHU-Route 2.0	BGR	MiniDeviation	NTHU-Route 2.0	BGR
bigblue4	102	138	36	64.71%	73.91%
newblue4	158	180	144	8.86%	20.00%
newblue5	2	0	0	100.00%	0.00%
Avg	87	106	60	57.86%	31.30%

TABLE III: Comparison with NTHU-Route 2.0 and BGR

Benchmark	MiniDeviation			NTHU-Route 2.0			BGR		
	TWL (e5)	TWD	CPU (Min)	TWL	TWD	CPU	TWL	TWD	CPU
adaptec1	63.3	88240	29.2	61.0	219704	8.2	62.7	109882	11.2
adaptec2	63.7	64534	5.1	62.8	129494	2.1	63.4	95914	2.8
adaptec3	147.4	25972	7.1	146.5	109754	5.1	146.9	42954	5.0
adaptec4	136.2	14266	4.2	135.4	88760	2.2	135.7	41776	2.1
adaptec5	167.2	46546	25.3	165.4	234564	13.3	167.0	83172	16.7
bigblue3	142.7	13506	7.4	142.2	47912	4.0	142.3	18218	3.5
bigblue4	239.4	38852	76.3	240.5	206684	13.8	241.6	49182	14.8
newblue2	85.4	10626	0.8	84.9	31544	0.6	85.2	19148	0.7
newblue4	140.5	49790	110.5	139.6	328758	65.0	140.3	62060	28.1
newblue5	246.3	63422	23.9	245.0	200620	15.0	246.0	78216	14.5
newblue6	198.6	105624	78.3	189.4	317738	19.0	197.5	185900	30.7
Ratio	1	1	1	0.99	4.05	0.49	1	1.61	0.51

We then compare MiniDeviation with NTHU-Route 2.0 [3] and BGR [10] in terms of the total overflow (TOF), total wirelength deviation (TWD), runtime (CPU) and total wirelength (TWL). We used the published results in their papers. The results are shown in Table II and Table III. The total overflow comparison only involves the benchmarks with overflow.

In terms of total overflow, MiniDeviation and BGR can complete 9 of 11 benchmarks without causing any overflow. However, NTHU-Route 2.0 cannot solve one of the 9 benchmarks. In particular, MiniDeviation obtains the best routing solutions with lowest overflow for the benchmarks with overflow in Table II, which is one of the most important indicators for evaluating the quality of routing solution. Especially for bigblue4, MiniDeviation not only greatly reduces the total overflow but also yields the least wirelength. It is due to that the iterative use of multi-stage double maze routing strategy to average congestion and adjust routing resource. In addition, the correct sequence of maze routing has a great effect on coordinating routing resource and reducing overflow, which is employed during post-routing stage. For the difficult instance newblue4, reduction in overflow is also obvious. We believe that MiniDeviation can do best than the other two routers for any benchmarks with overflow. Compared with NTHU-Route 2.0 and BGR, the experimental results show that MiniDeviation achieves 57.86% and 31.30% less total overflow for benchmarks with overflow, respectively.

As Table III shows, since MiniDeviation needs to match the bus wirelength by detouring, MiniDeviation is slightly worst in terms of total wirelength (TWL). However, MiniDeviation can achieve the best performance on minimizing the total wirelength deviation (TWD) by considering deviation cost in each stage. On average, total wirelength deviation of MiniDeviation is 4.05 and 1.61 times better than NTHU-Route 2.0 and BGR. In addition, the runtime ratio among MiniDeviation, NTHU-Route 2.0 and BGR is 1:0.49:0.51.

V. CONCLUSION

In order to cope with length-matching of buses in global routing, this paper presented a global router called MiniDeviation, characterized by considering wirelength deviation in global routing and equipped with several efficient techniques. Experimental results demonstrate the good performance of MiniDeviation in terms of total wirelength deviation and total overflow. Compared with NTHU-Route 2.0 and BGR, MiniDeviation shows a huge advantage. Future work would be considering timing flexibility on the circuit level [14]-[15] to enable more freedom in global routing.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grants No.61877010 and No.11501114, and the Fujian Natural Science Funds under Grant No.2019J01243. Genggeng Liu is the corresponding author of the article.

REFERENCES

[1] M. Pan, Y. Xu, Y. Zhang and C. Chu, "FastRoute: An Efficient and High-quality Global Router," *VLSI Design*, vol. 2012, pp. 1-18, 2012.

[2] W. Liu, W. Kao, Y. Li and K. Chao, "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing With Bounded-Length Maze Routing," *IEEE TCAD*, vol. 32, no. 5, pp. 709-722, 2013.

[3] Y. Chang, Y. Lee, J. Gao, P. Wu and T. Wang, "NTHU-Route 2.0: A Robust Global Router for Modern Designs," *IEEE TCAD*, vol. 29, no. 12, pp. 1931-1944, 2010.

[4] H. Chen, C. Hsu and Y. Chang, "High-Performance Global Routing with Fast Overflow Reduction," in *Proc. ASP-DAC*, 2009, pp. 582-587.

[5] F. Mo and R. K. Brayton, "Semi-Detailed Bus Routing with Variation Reduction," in *Proc. ISPD*, 2007, pp. 143-150.

[6] T. Yan and M. D. F. Wong, "BSG-Route: A Length-Constrained Routing Scheme for General Planar Topology," *IEEE TCAD*, vol. 28, no. 11, pp. 1679-1690, 2009.

[7] R. Zhang, T. Pan, L. Zhu and T. Watanabe, "A Length Matching Routing Method for Disordered Pins in PCB Design," in *Proc. ASP-DAC*, 2015, pp. 402-407.

[8] C. Hsu, S. Hung, H. Chen, F. Sun and Y. Chang, "A DAG-Based Algorithm for Obstacle-Aware Topology-Matching On-Track Bus Routing," in *Proc. DAC*, 2019, pp. 217:1-217:6.

[9] J. Chen, J. Liu, G. Chen, D. Zheng and F. Y. E. Young, "MARCH: MAze Routing Under a Concurrent and Hierarchical Scheme for Buses," in *Proc. DAC*, 2019, pp. 216:1-216:6.

[10] P. Liao and T. Wang, "A Bus-Aware Global Router," in *Proc. SASIMI*, 2018, pp. 20-25.

[11] C. Chu and Y. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *IEEE TCAD*, vol. 27, no. 1, pp. 70-83, 2008.

[12] R. Kastner, E. Bozorgzadeh and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *IEEE TCAD*, vol. 21, no. 7, pp. 777-790, 2002.

[13] W. Liu, Y. Li and C. Koh, "A Fast Maze-Free Routing Congestion Estimator With Hybrid Unilateral Monotonic Routing," in *Proc. ICCAD*, 2012, pp. 713-719.

[14] L. Zhang, B. Li and U. Schlichtmann, "PieceTimer: A holistic timing analysis framework considering setup/hold time interdependency using a piecewise model," in *Proc. ICCAD*, 2016, pp. 100:1-100:8.

[15] L. Zhang, B. Li, M. Hashimoto and U. Schlichtmann, "VirtualSync: Timing Optimization by Synchronizing Logic Waves with Sequential and Combinational Components as Delay Units," in *Proc. DAC*, 2018, pp. 26:1-26:6.