Fashion: A Fast and Accurate Solution to Global Routing Problem

Zhen Cao, Tom Tong Jing, *Member, IEEE*, Jinjun Xiong, *Member, IEEE*, Yu Hu, *Student Member, IEEE*, Zhe Feng, Lei He, *Member, IEEE*, and Xian-Long Hong, *Fellow, IEEE*

***Abstract*—This paper presents a fast and accurate solution, namely Fashion, to routability-driven global routing problem. Fashion is based on two efficient yet effective techniques: 1) dynamic pattern routing (DPR) and 2) movable-segment-driven DPR. These two techniques enable Fashion to explore large solution space to achieve high routability with low time complexity. Compared with BoxRouter, Fashion has a shorter wire length and reduces overflow and runtime by 5 and 15 times, respectively. Compared with FastRoute, Fashion has similar runtime but 90% smaller overflow and 1.9% shorter wire length. Fashion is significantly better than Labyrinth and Fengshui in terms of overflow, wire length, and runtime.**

***Index Terms*—Flexibility, global routing, physical design, routability, Steiner tree.**

I. Introduction

THE SUCCESS of future integrated circuit designs requires the consideration of physical impact. Routability-aware global routing is important to achieve this goal and is useful in physical synthesis. Many existing global routing algorithms focus on congestion reduction [1]–[10]. In recent publications, Labyrinth [5], [6] applied pattern-based technique to reduce congestion and wire length, SSTT [7] employed an efficient search space traversing technology for more optimized solution, and Fengshui [8] proposed the concept of amplified congestion estimation to reduce both overcongestion and wire length. More recent progress of global router includes integer-linear-programming-based BoxRouter [9], which has short wire length, low congestion, and a little speedup compared with that in [5] and [8]. Then, a trunk-decomposition-based global routing algorithm [11] was proposed. Compared with the algorithm in [5], this trunk-decomposition-based global routing algorithm [11] reduced more overflow but needed additional running time. Also, a fast routing algorithm, FastRoute, was developed to provide a quick but accurate wire length estimation in the process of placement [10]. FastRoute 2.0 [12] improved the routing quality of FastRoute based on monotonic routing and multisource maze routing.

Steiner tree algorithms [13]–[15] can be used to improve the routability. References [13] and [14] presented the idea of solving Steiner tree problem based on a precomputed lookup table. Kastner *et al.* [15] proposed the concept of flexibility in the rectilinear Steiner minimal tree (RSMT) problem. These tree algorithms are helpful, but the final congestion reduction still mainly relies on an effective global routing algorithm.

The problem of routability-aware global routing, however, is far from being solved. This paper focuses on the global routing problem for congestion reduction and algorithm efficiency. The major contributions of this paper are as follows.

1) An efficient dynamic pattern routing (DPR) technique to achieve the optimal routing solution for two-pin nets. This technique is flexible, and it can be extended to consider via minimization for design for manufacturability (DFM).
2) A movable-segment-driven DPR technique to efficiently explore routing solutions for less congestion.
3) A routability-aware global routing algorithm, which is called Fashion, with cost functions for best tree selection to achieve high speed and high quality of optimization.

We compare Fashion with published global routers. Compared with Labyrinth [5] and Fengshui [8], Fashion simultaneously reduces overflow and wire length with up to 59-times runtime reductions. Compared with BoxRouter [9], Fashion reduces wire length and overflow with 15-times runtime reduction. Compared with FastRoute [10], Fashion has a similar runtime but has 90% smaller overflow and 1.9% shorter wire length.

The rest of this paper is organized as follows. The problem formulation and some basic definitions are introduced in Section II. The DPR technique is proposed in Section III. In Section IV, we present the movable-segment-driven DPR technique. Fashion routing algorithm is described in Section V. Experimental results are presented in Section VI, and Section VII concludes this paper.

Manuscript received January 14, 2007; revised May 18, 2007. This work was supported in part by the Key Project of Chinese Ministry of Education under Grant 106008, in part by the Specialized Research Fund for the Doctoral Program of Higher Education (SRFDP) of China under Grant 20050003099, and in part by the National Natural Science Foundation of China (NSFC) under Grant 90607001. This paper was recommended by Associate Editor L. Sheffer.

Z. Cao and Z. Feng were with the Computer Science and Technology Department, Tsinghua University, Beijing 100084, China. They are now with the Electrical Engineering Department, University of California, Los Angeles, CA 90095 USA (e-mail: caoz@ucla.edu; feng07@ucla.edu).

T. T. Jing, Y. Hu, and L. He are with the Electrical Engineering Department, University of California, Los Angeles, CA 90095 USA (e-mail: tomjing@ucla.edu; hu@ee.ucla.edu; lhe@ee.ucla.edu).

J. Xiong was with the Electrical Engineering Department, University of California, Los Angeles, CA 90095 USA. He is now with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: jinjun@us.ibm.com).

X.-L. Hong is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: hxl-dcs@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TCAD.2008.917590

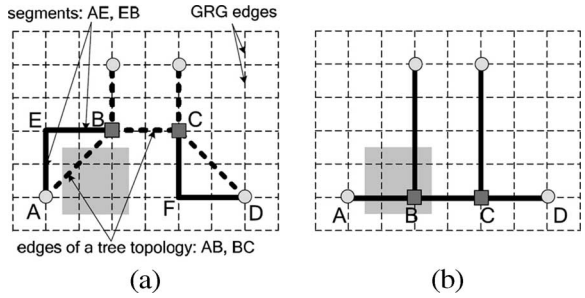


Fig. 1. RST with or without flexible edges (gray area indicates congestion). (a) Two flexible edges, AB and CD, in a tree. (b) No flexible edge in a tree.

II. Preliminaries

A. Problem Formulation

Global routing problem is often formulated by partitioning the routing area into global cells and mapping all physical pins in each cell into the center of the cell. The centers are vertices in the global routing graph (GRG), and GRG edges are added to adjacent vertices. We introduce the following concepts in the context of routability-driven global routing.

Given c_e as the capacity of a GRG edge e and d_e as the routing demand for edge e, the overflow of edge e is defined as follows:

$$overflow_e = \begin{cases} d_e - c_e, & \text{if } d_e > c_e \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The total overflow of the entire routing area *tof* is as follows:

$$tof = \sum_{e \in E} overflow_e. \quad (2)$$

B. Basic Definitions

Segment: A segment is a horizontal or vertical edge connecting two vertices in GRG, which consists of either one or more than one GRG edge. As shown in Fig. 1(a), AE and EB are segments. Both of them consist of two GRG edges.

Tree Topology: For a given rectilinear Steiner tree (RST), there is a corresponding graph $G(T, E)$ called the tree topology, where T is the set of pins and Steiner points, and E is the set of edges connecting the vertices in T. Note that the topology of an RST just describes the abstract connection between the pins and Steiner points. It does not describe the real GRG edge assignment of this RST. A tree topology is shown in Fig. 1(a) in dashed lines.

Routing Solution of a Tree Topology: Routing solution is the real GRG edge assignment of its tree topology given by the global routing algorithm. For a given tree topology, there may exist many routing solutions, as shown in Fig. 2, where (a) is the tree topology and we use beeline to indicate edge, and (b) and (c) are two valid routing solutions.

Edge of a Tree Topology: An edge of a tree topology is the connection between two vertices of this tree topology, such as dashed lines AB and CD shown in Fig. 1(a). Note the difference between an edge of a tree topology and an edge on GRG. Routing solution of an edge of a tree topology consists of one or more GRG edges. As shown in Fig. 1(a), A and D are pins, B and C are Steiner points, and routing solution AEB (solid line) of edge AB (dashed line) of the tree topology consists of four GRG edges.

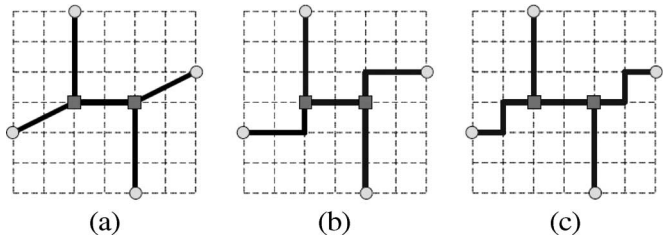


Fig. 2. (a) Tree topology. (b) and (c) Routing solutions.

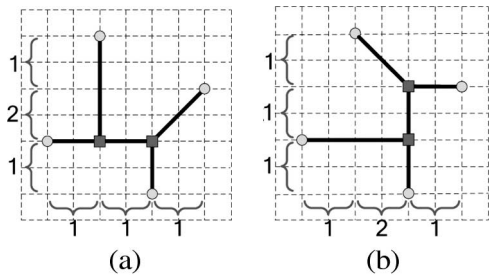


Fig. 3. Different POWVs for a given four-pin net. (a) POWV (1, 1, 1, 1, 2, 1). (b) POWV (1, 2, 1, 1, 1, 1).

Flexibility: Same as in [15], a flexible edge of a tree topology allows more than one minimum length routing solution. As shown in Fig. 1(a), the tree has two flexible edges (AB and CD), whereas the tree in Fig. 1(b) has no flexible edge. Flexible edges leave more freedom for global routing than nonflexible edges do. Thus, the tree in Fig. 1(a) can avoid congested area (gray color) by exploiting flexible edges to achieve the minimum length, whereas the tree in Fig. 1(b) cannot. Therefore, we would like to exploit such flexibility to obtain better routing solutions.

C. Flute Technique

Reference [14] presented a fast and accurate RSMT algorithm based on a lookup table, which is called FLUTE. In FLUTE, the set of all degree n nets is partitioned into $n!$ categories according to their relative pin locations. For every category, several potentially optimal wire length vectors (POWVs, i.e., different linear combinations of distances between adjacent Hanan grid lines [16]) are stored in a lookup table to give a fast estimation of minimal wire length. As shown in Fig. 3, two POWVs, (1, 1, 1, 1, 2, 1) and (1, 2, 1, 1, 1, 1), for one category of four-pin net are stored in a lookup table. The minimal wire length can be estimated by calculating the wire length considering the real distance between adjacent Hanan grid lines. For high-degree nets, without making the table size impractically large, a net-breaking technique is used to divide the net into subnets recursively so that the wire length of each subnet can be looked up from the table. Hence, it only finds an approximated minimal wire length for high-degree nets.

For every POWV, there is one tree topology stored in FLUTE, as shown in Fig. 3(a) and (b), respectively. However, in global routing, only one tree topology is not enough for

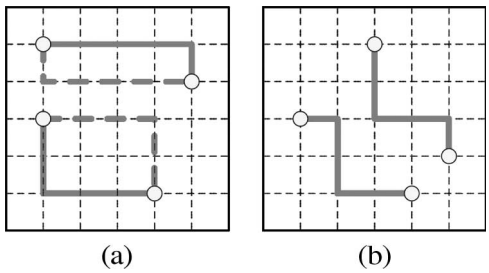


Fig. 4. L- and Z-shape PRs. (a) L-shape. (b) Z-shape.

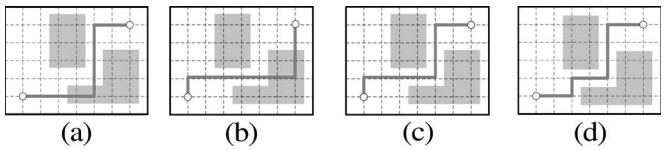


Fig. 5. PR in congested area. (a) Z-shape. (b) Z-shape. (c) Three-bend. (d) Four-bend.

avoiding congested area. Hence, we present a fast search technique to consider all possible routing solutions for every POWV, which will be introduced in Sections III and IV.

III. DPR

In global routing, researchers often decompose a tree topology into two-pin nets. Then, they use pattern routing (PR) (L- or Z-shape) or maze routing to route each two-pin net separately. However, the searching space of L- or Z-shape PR is small, and the time complexity of maze routing is high. Here, we present a new and flexible technique to connect a two-pin net, which is called the DPR. Compared with L- or Z-shape PR, this technique can explore patterns with more than two bends, enabling us to search larger solution space with the same minimal wire length and low time complexity.

A. PR

As shown in Fig. 4, the L-shape (one-bend) PR only allows routing on the bounding box, whereas Z-shape (two-bend) PR only allows routing with two bends on or inside the bounding box. Compared with maze routing, the time complexity of L- or Z-shape PR is much lower. However, in the routability-driven routing problem, particularly in the dense cases, L- or Z-shape PR cannot avoid some congested area, as shown in Fig. 5(a) and (b). Therefore, new fast PR technique with more than two bends is needed for routability-driven routing, as shown in Fig. 5(c) and (d).

B. DPR Technique

We present a technique called the DPR to search for more than two-bend routings with low time complexity and to keep the optimality of minimal wire length due to the property of dynamic programming.

Theorem 1: There are $C(m+n, m)$ different routing solutions to connect an edge with minimal wire length, where m and n are the vertical and horizontal distances of the edge, respectively.

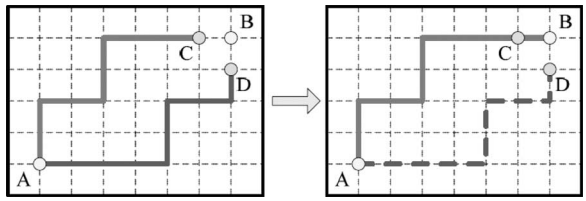


Fig. 6. DPR.

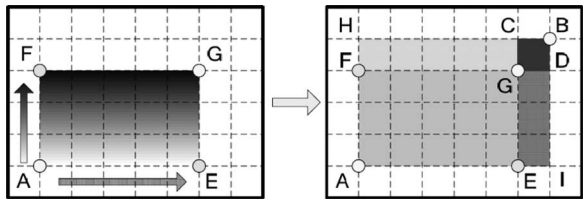


Fig. 7. Routing and incremental routing by DPR.

Proof: Every routing solution can be formulated as choosing m vertical steps out of total $m+n$ steps; hence, there are a total of $C(m+n, m)$ different combinations. ■

We employ linear functions to estimate the routing cost of every edge of GRG, which will be discussed in detail in Section V-B. The routability-driven routing problem is then formulated to find the routing solution with a minimal cost. Such a problem is solved optimally by combining dynamic programming. The key of dynamic programming is that the subsolution of the optimal solution must be optimal too. To show this, we have the following theorem.

Theorem 2: With properly defined linear cost functions, the optimal routing solution must be optimal in its subroutings.

Proof: (By way of contradiction) Suppose that the routing with a minimal cost has a subrouting that is not optimal. Then, this subrouting can be replaced with another routing with a lower cost. Since the cost function is linear, the so-obtained new routing would have a lower cost than before, which is contradictory to the assumption that the original solution is optimal. Therefore, all subroutings of the optimal routing must be optimal too. ■

With Theorem 2, we can utilize dynamic programming to find the optimal routing for a two-pin net. The idea is shown in Fig. 6. To find an optimal routing from A to B, we only need to find the optimal routings from A to C and from A to D. Then, by adding the routing cost for CB and DB, we can choose the optimal routing from A to B out of the two candidates, which may be a routing path ACB, as shown in Fig. 6.

In Fig. 7, the DPR first finds optimal routings along line AE, then along line AF just above AE, until it finally reaches G. The pseudocode of DPR is shown in Fig. 8.

Theorem 3: The time complexity of DPR, for $m \times n$ GRG grids within the bounding box of a two-pin net $n = \{(x1, y1), (x2, y2)\}$, is $O(mn)$ that is the same as that of Z-shape PR.

Proof: Given a two-pin net $n = \{(x1, y1), (x2, y2)\}$ with $m \times n$ GRG grids, let Z be the segments on and within the bounding box of n, and let L be the segments on the bounding box of n. Then, $|Z| = m \times (n+1) + n \times (m+1) = 2mn + m + n$. $|L| = (m+n) + (m+n) = 2(m+n)$. Thus,

Algorithm 1: DPR
Input: Routing graph G, a two-pin net $n = \{(x1, y1); (x2, y2)\}$
Output: Optimal routing solution for net n

1. **if** x_1 equals to x_2, or y_1 equals to y_2 **then**
2. return the straight line routing from $(x1; y1)$ to $(x2; y2)$;
3. **end if**;
4. **for** dy from 0 to $(y_2\text{-}y_1)$ **do**
5. **for** dx from 0 to $(x_2\text{-}x_1)$ **do**
6. current position is $(x_1{+}dx, y_1{+}dy)$, namely p;
7. get two points and corresponding segments left and below p;
8. calculate the corresponding cost for routings from (x_1, y_1) to p that pass these two points;
9. choose one with lower cost as parent of p, record the cost;
10. **end for**;
11. **end for**;
12. back trace from (x_2, y_2) to (x_1, y_1), get the routing solution s;
13. **return** s;

Fig. 8. Pseudocode of DPR algorithm.

$O(Z\text{-shape}) = O(mn)$ and $O(L\text{-shape}) = O(m + n)$. That is, for L-shape PR, time complexity is $O(m + n)$, whereas for Z-shape PR, it is $O(mn)$. With analysis of DPR algorithm, we observe that each segment on or within the bounding box is visited only once. Therefore, the time complexity of DPR is $O(mn)$ that is the same as that of the Z-shape PR. ■

DPR is powerful for routability-driven global routing because, compared with Z-shape routing with the same time complexity, the searching space of DPR routing is larger. DPR obtains the optimal routing solution with a minimal length by considering all possible $C(m + n, m)$ minimal wire-length routing solutions, whereas Z-shape only considers $m + n$ routing solutions.

DPR enables a pervasive technique for connecting a two-pin net. Many global routing algorithms decompose a multipin net into several two-pin nets and route every two-pin net. DPR can be adopted in such cases. The low time complexity of DPR enables these algorithms to optimize routing more effectively and efficiently.

DPR technique is also inheritable due to dynamic programming, which enables us to solve the routing problem shown in Fig. 7. Suppose that we have found the optimal routing solution from A to G, which means that we have already known the optimal routing solutions from A to any point in AEGF. When we extend the problem to connecting A and B, we just need to find optimal routing solutions from A to C and from A to D, respectively, based on the optimal routing solutions that we have known. Then, with adding additional routing cost from C to B and from D to B, respectively, we choose the better one from A → C → B and A → D → B and get the optimal routing solution from A to B. In this way, we save runtime.

IV. Movable-Segment-Driven DPR

A. Motivation

As discussed in Section II-C, only one topology for each POWV is stored in FLUTE, which is not sufficient for global routing. When using DPR to find the best routing solution of one topology under the constraint of minimal wire length, we can explore more routing solutions by making use of flexible edges in the stored tree topology. However, for nonflexible edges, we have to connect its end vertices directly. Thus, the search space explored is still limited. Therefore, in the following, we propose the movable-segment-driven DPR technique to search for more routing solution space by combining those so-called movable segments.

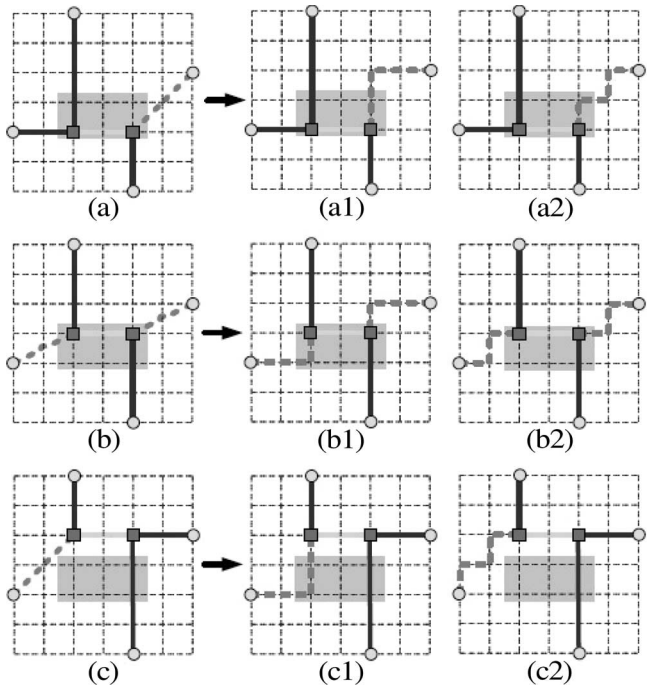


Fig. 9. Different tree topologies and routing solutions for one POWV.

This idea is shown in Fig. 9, where the red (dark color) square points are Steiner points, the dotted edge indicates that this edge is a flexible one, the green (light color) edge indicates that this edge is a movable segment, and the gray area indicates congestion (we use beeline to indicate connection). Fig. 9(a) shows the original tree topology for POWV (1, 1, 1, 1, 2, 1) stored by FLUTE. If sticking with flexible edges, we can only search the limited solution space even by using DPR to explore different routing solutions for this topology, with two of them shown in Fig. 9(a1) and (a2), respectively. This is not enough for avoiding congestion. We observe that all five edges of the tree are still involved in the congested area.

Note that some of other tree topologies are transformable from the original one by moving the green segment up, with two of them shown in Fig. 9(b) and (c), respectively. Then, DPR searching for each flexible edge in Fig. 9(b) and (c) can cover other solution space of this POWV. Four routing solutions of the search results are shown in Fig. 9(b1), (b2), (c1), and (c2), respectively. Finally, the routing solution with most congestion reduction is found, in which only one edge of the tree is involved in the congested area, as shown in Fig. 9(c2).

B. Identification of Movable Segment

The concept of movable segment/edge in a routing tree is introduced in [15]. Here, with a detailed study about all the

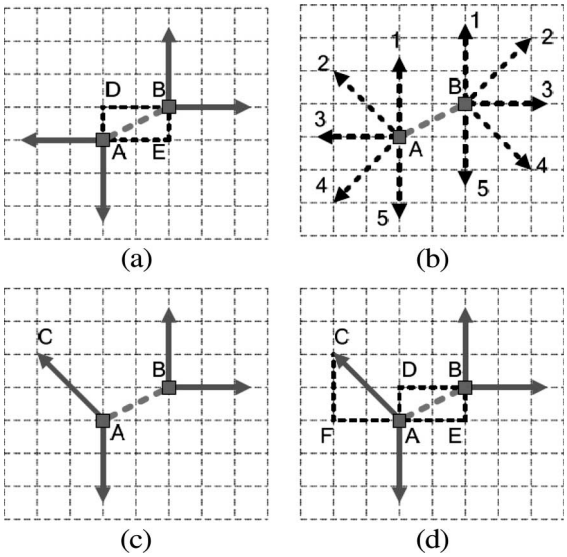


Fig. 10. Possible flexible edges connected by vertices with a degree of three or four. (a) Flexible edge AB. (b) Some possible directions. (c) Non-flexible edge AB. (d) Duplicated edges.

possible types of edges in a tree topology, we propose to identify movable segments as follows: First is to classify movable segments, and then, second is to serve movable-segment-driven DPR technique.

Theorem 4: Under the constraint of minimal wire length, edges connected by two vertices with a degree of three or four cannot be flexible, except that the topology is similar to that in Fig. 10(a).

Proof: Assume that the edge connected by vertices with a degree of three or four is flexible. Without loss of generality, we assume that these two vertices are located in directions shown in Fig. 10(b). We label some possible directions to adjacent vertices as shown in the figure (numbered from 1 to 5).

Then, only one tree topology is possible under the constraint of minimal wire length, which is shown in Fig. 10(a). That is, choosing the third and the fifth directions of the left vertex and the first and the third directions of the right one, from which AB is flexible since AB can route as ADB or AEB. Otherwise, as shown in Fig. 10(c), the wire length is not minimum because of unnecessarily duplicated edges to keep A and B as Steiner points. That is, no matter how AB routes, such as ADB or AEB, or other routes, as shown in Fig. 10(d), they are partially duplicated in the vertical direction such as AD or EB, which violates the minimal length constraint. ■

Theorem 5: Only nonflexible edges can be moved under the constraint of minimal wire length.

Proof: With Theorem 4, flexible edges are either like type shown in Fig. 10(a) or connected by at least one vertex with less than degree of three.

For a flexible edge shown in Fig. 10(a), it is impossible to move one of the endpoints of flexible edge without increasing the wire length.

For other flexible-edge cases connected by less than three-degree vertices, at least one of the endpoints of flexible edge is not Steiner point because the degree of Steiner point should be more than two. It cannot be moved without adding a new vertex and increasing the wire length. For example, in Fig. 11, the movement of non-Steiner vertex A adds new vertex A1 and increases wire length by $|AA1|$. Therefore, only segments (i.e., nonflexible edges) can be moved. ■

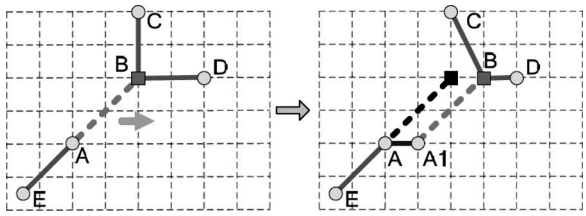


Fig. 11. Impact of flexible-edge movement.

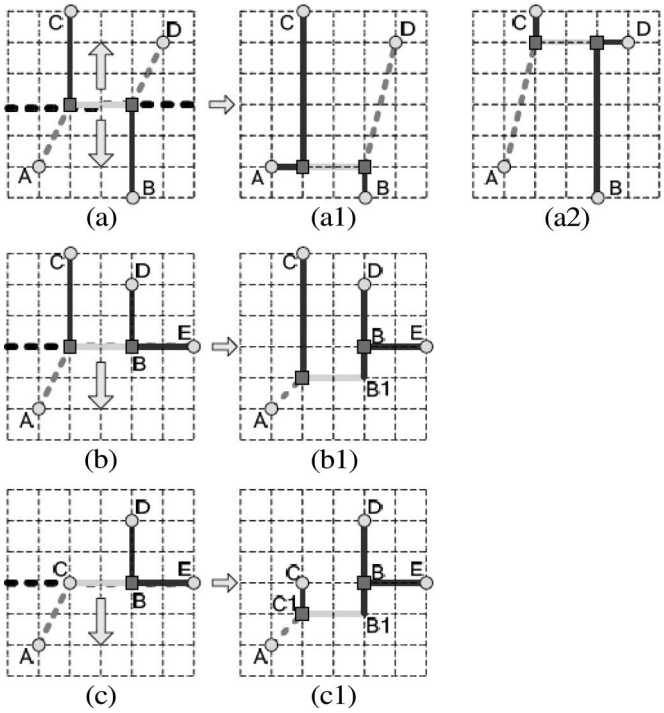


Fig. 12. Movement of segment.

Theorem 6: Under the constraint of minimal wire length, a segment is movable if and only if its two vertices have adjacent vertices on the same side of the segment.

Proof: If a segment has two adjacent vertices on the same side, it can be moved toward them. As shown in Fig. 12(a), the adjacent two vertices of a segment, A and B (or C and D), are on the same side of it. Then, the segment can be moved upward or downward. The maximum movement distance is limited by the location of one vertex closer to the segment, i.e., A or D, as shown in Fig. 12(a1) and (a2), respectively. If a segment does not have two adjacent vertices on the same side, there are two situations. If the endpoints of the segment are both Steiner points, as shown in Fig. 12(b), the movement adds new vertex B1 and causes an increment of wire length by $|BB1|$, as shown in Fig. 12(b1). If one of the endpoints of the segment is a non-Steiner point, as shown in Fig. 12(c), C is a non-Steiner point, then the movement adds two new vertices C1 and B1 and causes the increment of wire length by $|BB1|$, as shown in Fig. 12(c1). ■

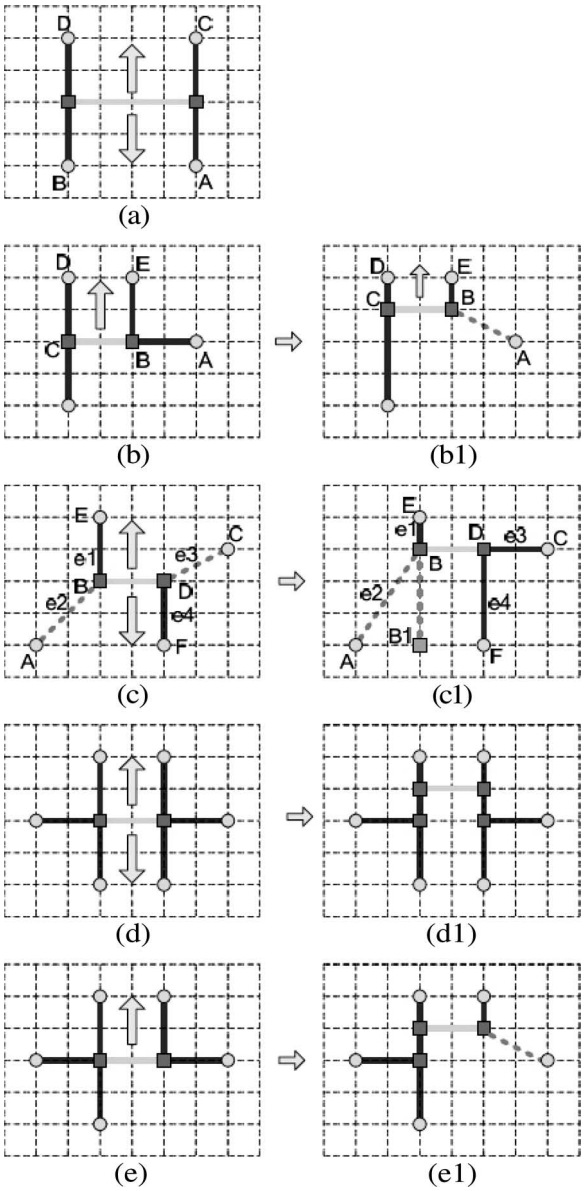


Fig. 13. Different types of moveable edges. (a) Type A. (b) and (c) Type B. (d) and (e) Type C.

C. Three Types of Movable Segment

With Theorems 4–6, we can identify movable segments. We classify movable segments into three types based on their characteristics shown in Fig. 13.

Type A) The segment movement does not influence the flexibility of adjacent edges, which is shown in Fig. 13(a).

Type B) The segment movement increases or decreases the flexibility of adjacent edges, as shown in Fig. 13(b) and (c), respectively. In Fig. 13(b), an upward movement makes edge BA become flexible, which is shown in Fig. 13(b1). In Fig. 13(c), an upward movement of the segment decreases the flexibility of edge CD, increases the flexibility of AB, and finally makes CD become a segment [see Fig. 13(c1)]. A downward movement decreases the flexibility of AB and increases the flexibility of CD.

Type C) The movement of the movable segment causes the generation of new vertices. As shown in Fig. 13(d) and (e), respectively, the downward or upward movement of the movable segment causes the generation of at least one new Steiner point.

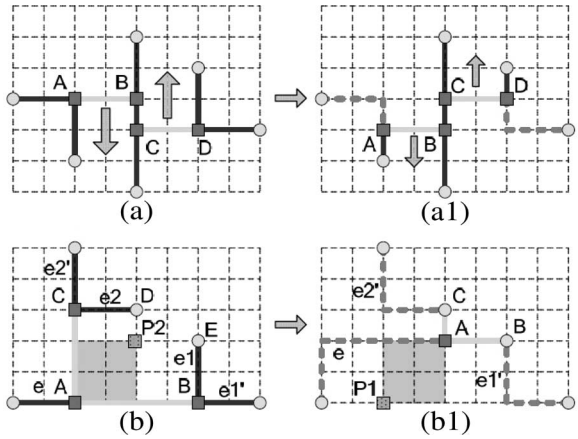


Fig. 14. Dependence of movable edges.

For Type A, we first find out the max movement range of the segment. We then calculate the corresponding cost at each possible position and choose the best one.

For Type B, DPR can be used to find the changed optimal routing solution incrementally for flexible edges due to the property shown in Fig. 7. For example, when the movement extends the flexibility of edge AB, as shown in Fig. 13(b1), we just use DPR to route the new flexible edge AB incrementally but do not need to route the edge ABE. Fig. 13(c) shows that two flexible edges are impacted by the movement. We also use DPR for each flexible edge. We assume that the endpoint of each flexible edge is on its maximum movable location. For example, location B shown in Fig. 13(c1) is the maximum movable location of the endpoint of AB. In this way, we can get the optimal routing of each possible location of B. That is, when moving the segment BD, the changed optimal routing solution of AB can be easily obtained based on the previous maximum-movable-location-related routing.

In Fig. 13(c1), B is in its maximum movable location. With DPR, we can easily find the optimal routing of AB at each location on BB1. Similarly, we can find the optimal routing of CD at each location on DF. Finally, we can obtain the optimal routing solution of segment BD based on the routes of AB and CD.

For Type C, we first add new Steiner point(s) and change the tree topology. We then convert the problem into either Type A or Type B. In Fig. 13(d1), after the upward movement of the segment, the segment-related part becomes Type A, whereas in Fig. 13(e1), after the upward movement of the segment, the segment-related part becomes Type B.

D. Dependence of Movable Segments

Sometimes, there is a dependence among movable segments. Thus, we cannot move them separately. There are two types.

Algorithm 2: Movable-segment driven DPR
Input: Topology T of a n-pin net n
Output: Optimal routing solution for net n

1. find all movable segments in T, treat each as a set, all in S;
2. merge two segments that are dependence type B into one set;
3. **for** each movable set s in S **do**
4. **if** s is dependance type B **then**
5. update topology by sub1(T, s);
6. **else if** s is movable segment type C **then**
7. add new vertexes into T;
8. update topology by sub2(T, s);
9. **else** update topology by sub2(T, s);
10. **end if**;
11. get routing solution r of T by DPR;
12. return r;

Algorithm sub1: DPR for dependent movable segments
Input: Topology T of a movable set s

1. see Figure 14(b) and Figure 14(b1), assume that the interconnected vertex is A, located at $P1$, segments are AB and AC, find maximal movable area for vertex A, which is the gray area, and also find the movable range of vertex C and vertex B, i.e., from vertex C to vertex D and from vertex B to vertex E;
2. use DPR to get costs of $e1$ and $e1$' when vertex B is at each possible position on BE;
3. use DPR to get costs of $e2$ and $e2$' when vertex C is at each possible position on CD;
4. for the adjacent edge of vertex A, e, use DPR to get cost of e when vertex A is at positions inside the moveable area;
5. get costs of AB and AC for all possible positions inside the movable area;
6. the total cost when vertex A is at each possible position can be calculated; choose the position with the minimal cost, then vertex A is at that position; update T;

Algorithm sub2: DPR for moveable segment
Input: Topology T of a movable set s

1. assume that the movable segment is BD, as shown in Figure 13(c), find maximal movable range for vertex B and vertex D, i.e., from vertex B to vertex $B1$ and from vertex D to vertex F in Figure 13(c1);
2. use DPR to get costs for $e1$ and $e2$ when vertex B is at each possible position;
3. use DPR to get costs for $e3$ and $e4$ when vertex D is at each possible position;
4. get the cost of BD at every possible position.
5. then the total cost at each possible position for BD can be calculated; choose the position with the minimal cost, then BD is at that position; update T;

Fig. 15. Pseudocode of movable-segment-driven DPR.

Type A is shown in Fig. 14(a). In this case, the movement of one segment impacts the tree topology. The tree topology has to be changed when the order of the two share-same-segment vertices (B and C) changes, as shown in Fig. 14(a1). Type A can be solved directly by Algorithm 2 (see Fig. 15) without any other specified procedures.

Type B is shown in Fig. 14(b). In this case, vertex A is the endpoint of two perpendicular-connected movable edges. Then, the movement of these two movable segments is dependent. In Fig. 14(b), the maximum movable location of vertex A is from P1 to P2 bounded in the gray area. When vertex A moves, both Steiner points, B and C, move too, as shown in Fig. 14(b1). We can also use DPR to find incrementally the new optimal routing solution.

E. Combining DPR With Movable Segment

Based on the above three types of movable segments and the dependence of movable segments, we design the algorithm of movable-segment-driven DPR. On the one hand, we need to find the movable segments of the original tree topology and move them to new locations to generate new tree topologies.

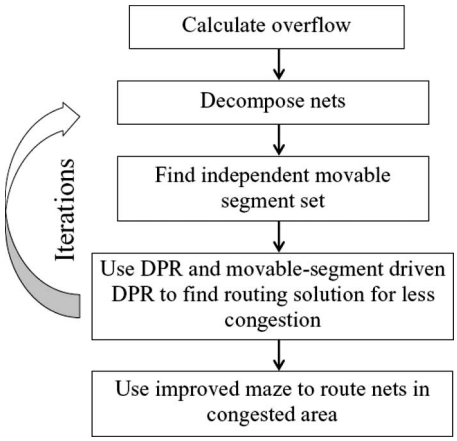


Fig. 16. Flowchart of Fashion algorithm.

Algorithm 3: Fashion algorithm
Input: Routing graph G and nets set N
Output: Routing solutions for every net in N

1. initial overflow calculation;
2. **for** every net list l in N **do**
3. get the POWVs set P of l;
4. **for** every POWV v in P **do**
5. use FLUTE to get the original topology t for vector v of net list l;
6. find independent movable segment sets M of t;
7. use movable-segment driven DPR to move segments in M to find the best routing solution s for v, put s in set S;
8. **end for**;
9. select the best solution s' in S, route it;
10. **end for**;
11. rip-up and reroute every net in congested area, using method the same of line 3-8, for several iterations;
12. rip-up and reroute every net in congested area, tree topology is selected using method of line 3-8, but maze routing is utilized in routing;
13. **return**;

Fig. 17. Pseudocode of Fashion algorithm.

On the other hand, we want to use the DPR technique to search edges to give the best routing for the new topologies. However, it is too costly if we use DPR on every topology separately. Therefore, we combine DPR with movable segments to reduce the time complexity. The detailed algorithm of movable-segment-driven DPR is shown in Fig. 15.

We perform the classification of movable segment types as follows. If a segment has two adjacent vertices on the same side of it, it can be moved. If a segment is movable and one of its endpoints has a degree of two or four, it belongs to type C. If two segments share one endpoint and they are perpendicular, they are dependence type B.

The maximal movement distance is the vertical or horizontal distance between the movable segment and the vertex that is one of the two same side vertices and is closer to the movable segment.

V. Global-Routing Algorithm Fashion

A. Main Flow of Fashion

The key problem in global routing is the generation of Steiner tree topology. A better Steiner tree topology can avoid congestion efficiently, reducing both congestion and total wire length. In addition, sometimes, the runtime of Steiner tree generation influences the runtime of global router severely. Our movable-segment-driven DPR and DPR techniques enable the global routing algorithm to generate Steiner tree topology by transformation from one, which saves runtime and explores more search space.

We present a global router based on the DPR and movable-segment-driven DPR techniques. The flowchart of Fashion is shown in Fig. 16. The pseudocode of Fashion is shown in Fig. 17.

B. Selection of Tree Routing Solution

We first find all movable segments and then classify them into independent movable sets. After that, we use DPR to move every segment in independent movable sets to minimize the total cost. The tree topology with the minimal cost is selected for the netlist to be routed. Two different cost functions are employed in different routing stages to select the best routing solution. The tree cost is the sum of the costs of all GRG edges in the routing tree.

In the earlier several times of iteration, we use cost function f_{earlier} on each GRG edge, which is defined as follows:

$$f_{\text{earlier}} = \alpha \times DisOv + \max(\gamma \times CurrRnet, \delta \times InitRnet) \tag{3}$$

where α, γ, and δ are weights, $DisOv$ is the difference between the routed nets and a discount (such as 70%) of the edge capacity, e.g., $DisOv =$ (the number of routed nets $- 70\% \times$ edge capacity), and $CurrRnet$ and $InitRnet$ represent the number of current routed nets and initial routed nets on the GRG edge, respectively.

In the later times of iteration, we use cost function f_{later} on each GRG edge, which is defined as follows:

$$f_{\text{later}} = \beta \times overflow_e + \gamma \times CurrRnet \tag{4}$$

where β is an empirical weight and $overflow_e$ represents the overflow of edge e [see (1)].

In the earlier several times of iteration, the congestion distribution in the chip area is quite uneven. That is, the congested areas are often concentrated, and other areas have more routing resources. We want to change such kind of situation quickly and greatly. Using f_{earlier} is helpful since it is a conservative estimation of congestion. However, in the later times of iteration, congestion distribution becomes even more than before; thus, we want to get the exact congestion information based on f_{later}.

To reduce runtime, we get a congestion information with cost functions f_{earlier} and f_{later} in the following way. We only choose the tree topology with minimal wire length from FLUTE and then route it by using DPR. After that, we record the routed net number on each GRG edge and get the routed net information, which is used for cost functions.

TABLE I
TESTCASES CHARACTERS

Circuits	Net#	Grids	Circuits	Net#	Grids
ibm01	13K	64 × 64	ibm06	34K	128 × 64
ibm02	19K	80 × 64	ibm07	46K	192 × 64
ibm03	26K	80 × 64	ibm08	49K	192 × 64
ibm04	31K	96 × 64	ibm09	59K	256 × 64
ibm05	30K	128 × 64	ibm10	66K	256 × 64

C. Rerouting and Net Ordering

In our algorithm, the initial routing solution is obtained by DPR and movable-segment-driven DPR techniques. After that, we first perform rip-up and rerouting also by using DPR and movable-segment-driven DPR techniques to reroute every congested net for some times of iteration. Then, we use maze routing. The motivation is that maze routing has a much higher time complexity and a larger searching space than DPR does. Therefore, we first use the fast method to reduce congestion while keeping a short wire length. Then, we use maze routing to reduce congestion. This strategy can reduce runtime while keeping high routability.

In our algorithm, we order the nets by the size of their bounding boxes. In the initial routing step, the order is from small to large, whereas in the rip-up and rerouting, the order is from large to small. The motivation is that a net with a bigger bounding box is more flexible than a net with a smaller one. Therefore, in initial routing, big ones are routed after the small ones to avoid congestion. In rerouting, the information of congestion is clear. Therefore, big nets should be rerouted first to quickly reduce congestion.

VI. EXPERIMENTAL RESULTS AND DISCUSSIONS

We implemented Fashion algorithm in C++ and tested our program on ISPD'98 benchmarks [17]. Table I shows the characteristics of all benchmarks. The experiments in this paper are performed in a 1.6-GHz Pentium-IV Linux desktop.

To demonstrate the effectiveness of our DPR technique in finding a better routing solutions, we first compare DPR with L- or Z-shape PR in terms of the total overflow *tof* and runtime. The initial routing tree topologies are all given by FLUTE with minimal wire length for all three PR techniques. We report the results in Table II. Compared with both L- and Z-shape PR techniques, DPR reduces more than 30% $((1.42-1)/1.42)$ overflow but with the same runtime.

We then compare our router with published academic routers in terms of the total overflow *tof*, total wire length *twl*, and runtime. We ran Labyrinth 1.1 [6] and Fengshui 5.1 (newest implementation of Chi dispersion router) [8]. Since BoxRouter [9] and FastRoute [10] were also compared with Labyrinth 1.1 and Fengshui 5.1 in their papers, we used the published results of BoxRouter and FastRoute in their corresponding papers as references. The runtimes of BoxRouter and FastRoute are scaled according to their comparison with Labyrinth and Fengshui for fair comparison. The results of wire length, overflow, and runtime do not include the comparison for ibm05, since it is a trivial case, and FastRoute does not report the test results for it. The results are shown in Tables III–V, respectively.

TABLE II
COMPARISON WITH L- AND Z-SHAPE PR TECHNIQUES

circuit	L-shape PR		Z-shape PR		DPR	
	tof	cpu (*s*)	*tof*	cpu (*s*)	*tof*	cpu (*s*)
ibm01	2637	0.21	2556	0.22	1995	0.23
ibm02	5353	0.57	5308	0.57	4529	0.57
ibm03	2345	0.52	2151	0.54	1118	0.56
ibm04	4755	0.44	4666	0.46	3114	0.46
ibm05	161	1.85	227	1.90	20	1.95
ibm06	5820	1.26	5606	1.30	3936	1.30
ibm07	4577	1.20	4115	1.27	2937	1.28
ibm08	5949	2.02	5828	2.05	3506	2.07
ibm09	8547	1.67	7309	1.84	4990	1.78
ibm10	8026	2.45	7216	2.61	5463	2.60
Total	48170	12.19	44982	12.76	31608	12.80
Average	1.52	0.95	1.42	1.00	1.00	1.00

In Table III, our Fashion router has the shortest wire length even though BoxRouter is similar to Fashion. Labyrinth is the worst, and on average, it produces 17.7% more wire length than Fashion. In terms of the total overflow, compared with Fashion, the overflows of other four routers, BoxRouter, FastRoute, Labyrinth, and Fengshui, are much worse, with 54- to 5-times overflow more than Fashion. Fashion has 90% less overflow than FastRoute. Regarding runtime comparison, FastRoute and Fashion can achieve more than 15-times speedup than the other three.

Finally, we compare Fashion with a very recent router, FastRoute 2.0 [12], published at the same conference with the conference version [18] of this paper. We also used the published results of FastRoute 2.0 from [12]. According to Table III, Fashion has shorter wire length than FastRoute 2.0. In Table IV, Fashion has a slightly smaller overflow than FastRoute 2.0. The runtime ratio in [12] between FastRoute 2.0 and FastRoute is 1:0.578. In Table V, the runtime ratio between Fashion and FastRoute is 1:0.48. Thus, the runtime ratio between FastRoute 2.0 and Fashion is 0.96:1.

VII. CONCLUSION

This paper has presented a fast global routing algorithm based on two novel techniques: 1) a DPR technique to achieve optimal routing solutions for an edge with low time complexity and 2) a movable-segment-driven DPR technique to search more solution space for routability-driven RST problem. Based on the two techniques, a global router, called Fashion, with cost functions has been developed, and we compared it with the published routers. Fashion has shorter wire length than BoxRouter with 5- and 15-times smaller overflow and runtime, respectively. Compared with FastRoute, Fashion has similar runtime but has 90% smaller overflow and 1.9% shorter wire length. Fashion is significantly better than Labyrinth and Fengshui in terms of overflow, wire length, and runtime. We also compared Fashion with FastRoute 2.0 published at the same conference with the conference version of this paper. Fashion has a shorter wire length and a slightly smaller overflow with a similar running time.

TABLE III
WIRE LENGTH COMPARISON

(*: not including ibm05 since FastRoute and FastRoute2.0 did not give related results.)

	Labyrinth	Fengshui	FastRoute	FastRoute2.0	BoxRouter	Fashion
ibm01	76517	66006	67128	68489	65588	65503
ibm02	204734	178892	179995	178868	178759	173140
ibm03	185116	152392	151023	150393	151299	147765
ibm04	196920	173241	172593	175037	173289	171090
ibm05	420583	412197	-	-	409747	410202
ibm06	346137	289276	285882	284935	282325	280487
ibm07	449213	378994	376835	375185	378876	371111
ibm08	469666	415285	412915	411703	415025	408107
ibm09	481176	427556	426471	424949	418615	417138
ibm10	679606	599937	599433	595622	593186	588122
Total*	3089085	2681579	2672275	2665181	2656962	2622463
Average	1.1779	1.0225	1.0190	1.0163	1.0131	1

TABLE IV
OVERFLOW COMPARISON

	Labyrinth	Fengshui	FastRoute	FastRoute2.0	BoxRouter	Fashion
ibm01	398	189	250	31	102	23
ibm02	492	64	39	0	33	2
ibm03	209	10	1	0	0	0
ibm04	882	465	567	64	309	72
ibm06	834	35	33	0	0	0
ibm07	697	309	18	0	53	0
ibm08	665	74	58	0	0	0
ibm09	505	52	28	3	0	0
ibm10	588	51	18	0	0	0
Total	5270	1249	1012	98	497	97
Average	54.33	12.88	10.43	1.01	5.12	1

TABLE V
RUNTIME COMPARISON

	Labyrinth	Fengshui	FastRoute	BoxRouter	Fashion
ibm01	35.1	25.00	0.61	13.74	2.27
ibm02	58.9	81.78	1.74	58.22	3.08
ibm03	63.7	61.77	1.07	29.66	1.89
ibm04	145.5	94.27	1.43	41.65	3.78
ibm06	171.6	131.78	2.25	54.29	5.14
ibm07	408.4	218.79	2.89	91.13	5.77
ibm08	417.5	199.04	3.17	163.01	7.32
ibm09	673.1	234.36	3.76	119.71	6.27
ibm10	789.6	385.83	5.47	172.35	11.22
Total	2763.40	1432.62	22.39	743.76	46.74
Average	59.12	30.65	0.48	15.91	1

APPENDIX
EXTENSION TO VIA MINIMIZATION

As the VLSI feature size continues to shrink, the number of vias becomes a critical issue, particularly considering DFM, because vias have a great impact on the circuit performance, layout size, and yield rate.

One of the motivations for PR is that a routing solution with less bends can reduce the number of vias. L- or Z-shape PR allows one or two bends; thus, the via number for each two-pin net is constrained by one or two. However, they are not flexible for routability-driven routing. In the following, we extend the DPR technique to consider via minimization problem in the context of routability-driven global routing.

We explain our idea by using an example shown in Fig. 18, where a two-pin net is shown with a source pin at vertex A and a sink pin at vertex B. Same as DPR, we record the optimal solution for all vertices within the bounding box from vertex A to vertex B. The search sequence is the same as the DPR algorithm shown in Fig. 8.

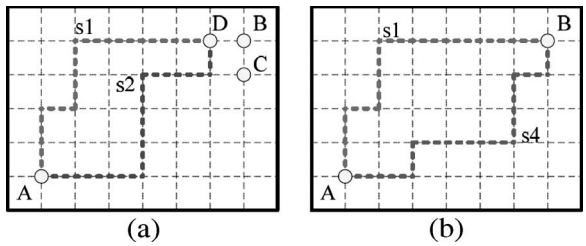


Fig. 18. DPR considering via minimization.

For simplicity of explanation, we focus on finding the optimal solution at any one vertex (e.g., vertex B). At each vertex, we record both the optimal routing solutions from the vertex to the left of and the vertex under this vertex. As shown in Fig. 18(a), we record the two optimal routing solutions for D, s1 from left and s2 from underside, and for C, s3 and s4 (not shown in the figure). When choosing the optimal routing solution coming to the vertex of B, both s1 and s2 are extended to B with a newly added cost, via number, i.e., we add via number as another component in the cost functions (3) and (4). Between the two extended solutions, s2 produces one more via at D by changing the routing direction from vertical to horizontal, whereas s1 introduces no additional via at D by keeping its horizontal direction. Because via number is considered as part of the cost function, by choosing the routing solution with lower cost (thus, s1 wins in this case), we explicitly take via minimization into account even in the context of routability-driven routing.

Similarly, we also can find the optimal routing solution coming from the vertex under B, i.e., s4 as shown in Fig. 18(b),

by comparing the cost between s3 and s4. Finally, we obtain the optimal routing solution from A to B, s1, by comparing the cost between s1 and s4.

It is easy to see that with the above extension, our DPR algorithm can obtain a smooth tradeoff between routability and via minimization, and the time complexity is the same as before.

ACKNOWLEDGMENT

The authors would like to thank Prof. C. Chu from Iowa State University for providing the source code of FLUTE.

REFERENCES

[1] M. Sarrafzadeh, C. Chiang, and C. Wong, "Global routing based on Steiner min-max trees," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 12, pp. 1318–1325, Dec. 1990.

[2] W. Swartz and C. Sechen, "A new generalized row-based global router," in *Proc. Des. Autom. Conf.*, 1993, pp. 491–498.

[3] J. Li, R. C. Carden, IV, and C. K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 208–216, Feb. 1996.

[4] C. Albrecht, "Provably good global routing by a new approximation algorithm for multicommodity flow," in *Proc. Int. Symp. Phys. Des.*, 2000, pp. 19–25.

[5] E. Bozorgzadeh, R. Kastner, and M. Sarrafzadeh, "Pattern routing: Use and theory for increasing predictability and avoiding coupling," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 7, pp. 777–790, Jul. 2002.

[6] *Labyrinth*. [Online]. Available: http://www.ece.ucsb.edu/~kastner/labyrinth

[7] T. Jing, X. Hong, H. Bao, J. Xu, and J. Gu, "SSTT: Efficient local search for GSI global routing," *J. Comput. Sci. Technol.*, vol. 18, no. 5, pp. 632–639, Sep. 2003.

[8] R. Hadsell and P. Madden, "Improved global routing through congestion estimation," in *Proc. Des. Autom. Conf.*, 2003, pp. 28–31.

[9] M. Cho and D. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," in *Proc. Des. Autom. Conf.*, 2006, pp. 373–378.

[10] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 464–471.

[11] D. Jariwala and J. Lillis, "Trunk decomposition based global routing optimization," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 472–479.

[12] M. Pan and C. Chu, "FastRoute 2.0: A high-quality and efficient global router," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 250–255.

[13] C. Chu, "FLUTE: Fast lookup table based wirelength estimation technique," in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 696–701.

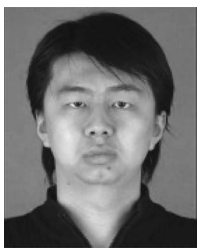
[14] C. Chu and Y. Wong, "Fast and accurate rectilinear Steiner minimal tree algorithm for VLSI design," in *Proc. Int. Symp. Phys. Des.*, 2005, pp. 28–35.

[15] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Creating and exploiting flexibility in rectilinear Steiner trees," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 5, pp. 605–615, May 2003.

[16] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl. Math.*, vol. 14, no. 2, pp. 255–265, Mar. 1966.

[17] C. Alpert, "The ISPD98 circuit benchmark suite," in *Proc. Int. Symp. Phys. Des.*, 1998, pp. 80–85.

[18] Z. Cao, T. Jing, J. Xiong, Y. Hu, L. He, and X. Hong, "DpRouter: A fast and accurate dynamic-pattern-based global routing algorithm," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2007, pp. 256–261.



Zhen Cao received the B.S. and M.S. degrees in computer science from Tsinghua University, Beijing, China, in 2005 and 2007, respectively. He is currently working toward the Ph.D. degree in electrical engineering at the University of California, Los Angeles.

His research interests include computer-aided design of VLSI circuits and systems, physical design, and computer architecture.

Tom Tong Jing (M'02) received the B.S. degree in electronic engineering and the M.S. and Ph.D. degrees in computer science from the Northwestern Polytechnical University, Xi'an, China, in 1989, 1992, and 1999, respectively.

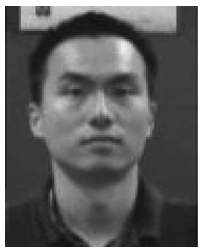
From September 1999 to April 2001, he was a Postdoctoral Researcher with the Computer Science and Technology Department, Tsinghua University, Beijing, China. He was a member of the faculty (Associate Professor from 2001 to 2004, Tenured Associate Professor from 2004 to 2006) with the Computer Science and Technology Department, Tsinghua University. He was a Visiting Scholar with the University of California, San Diego, and the Chinese University of Hong Kong, Hong Kong. He is currently a Research Associate with the Electrical Engineering Department, University of California, Los Angeles. He has authored or coauthored over 100 papers published in technical journals and conference proceedings. His research interests include electronic design automation, combinational optimization and algorithms, graph theory, and programming.

Dr. Jing has served as a TPC member of IEEE/ACM ASP-DAC 2006; the Secretary General and Chair of the Physical Design and Interconnect Optimization TPC-Subcommittee; the Session Chair of IEEE/ACM ASP-DAC 2005; the Session Chair of IEEE ASAP 2005; a TPC member, a Panel Speaker, and Session Cochair of IEEE ICCCAS 2004; a Session Chair of ISCI 2004; the Secretary General and TPC member of IEEE ASICON 2003; and the Session Cochair of IEEE/ACM ASP-DAC 2003. He was a recipient of an IEEE/ACM ASP-DAC Best Paper Award in 2005, ACM/IEEE ISQED Best Paper Nomination in 2005, and IEEE ASICON Outstanding Student Paper Award in 2003. He was the recipient of the Second Class Science and Technology Award from the Ministry of Education of China in 2005. He was the recipient of the First Class Award for Excellence in Teaching from the Beijing Municipal Education Commission in 2004 and the First Class Awards for Excellence in Teaching from Tsinghua University in 2002 and 2004.

Jinjun Xiong (S'04–M'06) received the B.E. (with honors) degree in precision instrument, the B.E. degree in industrial engineering, and the M.E. degree in precision instruments from Tsinghua University, Beijing, China, in 1998, 1998, and 2000, respectively. He received the M.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 2002, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles (UCLA), in June 2006.

He is currently a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His research interests include statistical timing analysis and optimization, design for manufacturability, design automation for VLSI circuits and systems, large-scale optimization, and combinatorial mathematics.

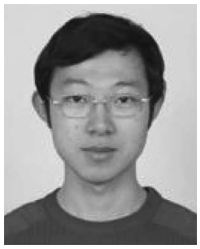
Dr. Xiong was the recipient of the 2005–2006 Outstanding Ph.D. Award in Electrical Engineering from UCLA, the Best Student Paper Award at the International Conference on ASIC 2003, and the Best Paper Award at the ACM International Symposium on Physical Design in 2006.



Yu Hu (S'05) received the B.S. and M.S. degrees in computer science from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently working toward the Ph.D. degree in the Electrical Engineering Department, University of California, Los Angeles (UCLA).

In the summer of 2006, he was with Xilinx Research Laboratory, San Jose, CA. He is currently a Graduate Student Researcher with the Electrical Engineering Department, UCLA. His current research interests include CAD for FPGA synthesis and ASIC physical synthesis. He is the author of over 20 technical papers in journals and international conferences and the holder of four patents in the field of CAD for VLSI designs.

Mr. Hu has been a full member of Sigma Xi since 2007. He received the Outstanding Graduate Student Award in 2005 from Tsinghua University.



Zhe Feng received the M.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2007. He is currently working toward the Ph.D. degree in the Electrical Engineering Department, University of California at Los Angeles, Los Angeles.

His research interest focuses on synthesis and physical design for FPGAs and on the physical design for ASICs.

Lei He (S'94–M'99) received the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), in 1999.

He was a faculty member at the University of Wisconsin, Madison, between 1999 and 2001. He also held visiting or consulting positions with Intel, Hewlett-Package, Cadence, Synopsys, Rio Design Automation, and Apache Design Solutions. He is currently an Associate Professor with the Electrical Engineering Department, UCLA. He has published over 150 technical papers. His research interests include VLSI circuits and systems, and electronic design automation.

Dr. He has been a technical program committee member for a number of conferences, including the Design Automation Conference, International Conference on Computer-Aided Design, International Symposium on Low Power Electronics and Design, and International Symposium on Field Programmable Gate Arrays. He received the National Science Foundation CAREER Award in 2000, the UCLA Chancellor's Faculty Career Development Award (highest class) in 2003, the IBM Faculty Award in 2003, the Northrop Grumman Excellence in Teaching Award in 2005, a Best Paper Award at the 2006 International Symposium on Physical Design, and multiple best paper nominations at the Design Automation Conference and International Conference on Computer-Aided Design.

Xian-Long Hong (M'95–SM'95–F'03) received the B.S. degree from Tsinghua University, Beijing, China, in 1964.

Since 1988, he has been a Professor with the Department of Computer Science and Technology, Tsinghua University. He has published five books and more than 400 papers. His research interests include very large scale integration layout algorithms and design automation systems.

Prof. Hong has served as a Steering Member of the Asia and South Pacific Design Automation Conference (ASPDAC) and the Cochair of the Technical Program Committee of ASPDAC in 1999, 2004, and 2005, respectively. He has been an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I since 2002.