Efficient Routability Checking for Global Wires in Planar Layouts

Naoyuki Iso Yasushi Kawaguchi Tomio Hirata

Faculty of Engineering
Nagoya University
Chikusa-ku, Nagoya-shi, Japan 464-01
Tel: +81-52-789-3440
Fax: +81-52-789-3089
e-mail: {fmiso, yasushi, hirata}@hirata.nuee.nagoya-u.ac.jp

Abstract— In VLSI and printed wiring board design, routing process usually consists of two stages: the global routing and the detailed routing. The routability checking is to decide whether the global wires can be transformed into the detailed ones or not. In this paper, we propose two graphs, the capacity checking graph and the initial flow graph, for the efficient routability checking.

I. Introduction

In VLSI and printed wiring board design, most of existing routers use a maze algorithm or a line search algorithm for finding a path in the routing area for each wire. Both algorithms find paths one by one. A major difficulty of such routing algorithms is that already routed wires are treated as fixed obstacles, so that routing for the next wire may be blocked by them. To overcome this difficulty, the global routing process has been proposed. Dai et al.[2] proposed a method for deciding whether every global wire can be transformed into a detailed wire or not. If the transfomation is possible, we can start the detailed routing process. Otherwise, we modify some global wires and check the routability again. Therefore, a fast routability checking is important.

In the case of planar layouts, we can check the routability of wires by comparing *capacity* with *flow* for every *cut*. Cut is a line segment connecting between two visible vertices in the routing area. Capacity of a cut is the maximum number of wires which can cross the cut. Flow of a cut is the number of wires passing across the cut. Cole and Siegel[1] showed that if the flow do not exceed the capacity for every cut then every global wire can be transformed into a detailed one. They presented a checking algorithm which runs in $O(n \log n)$ time, where n is the number of obstacles(terminals and modules) in the routing area. Their algorithm is too complicated and its implementation seems to be difficult. Leiserson and Maley[4] presented a simpler algorithm and it runs in $O(n^2 \log n)$ time.

In this paper, we propose the capacity checking graph(CCG) for an efficient routability checking. Using the CCG, we can test the routabilitiy in $O(n \log n)$ time on the average. Furthermore we propose the initial flow graph(IFG) for finding flows which are necessary for the use of the CCG.

The remainder of this paper is organized as follows. In Section II, we define our routing model. In Section III, we define the capacity checking graph for the routing model and present an algorithm for constructing it. We define the initial flow graph in section IV. In Section V, we describe the routing scheme using the CCG and IFG. Some concluding remarks are given in Section VI.

II. The Routing model

In this section, we describe our routing model. A routing area consists of a rectangle on a single layer and there is a rectlinear lattice on the rectangle. In the routing area, there are terminals, rectangular modules and wires. The wires interconnect two terminals without crossing each other and do not pass through the other terminals and modules. A terminal connects to the only one wire. Every terminal is placed at a lattice point. An obstacle area is the interior of a module. Outside of the routing area is also an obstacle area. Lattice points on the boundary of obstacles are called boundary points. We denote the set of boundary points by $\mathbf{V}_b$. $\mathbf{V}_t$ is the set of terminals in the routing area. Let $\mathbf{V}_{t+b}$ denote the union of $\mathbf{V}_t$ and $\mathbf{V}_b$. We denote by $\mathbf{V}_m (\subset \mathbf{V}_b)$ the set of corner points of modules and they are called module points. Let $\mathbf{V}_{t+m}$ denote the union of $\mathbf{V}_t$ and $\mathbf{V}_m$. Let $\mathbf{V}_{b-m}$ be the set of boundary points except the module points. In global routing, every wire is represented by its topology. In detailed routing, it is embedded on the lattice (See Figure 1). In the following, a "point" means a lattice point. Let a, b be two points in the routing area. When a line segment $\overline{ab}$ do not cross modules or terminals, we say that a and b are visible each other. We also say that the two points are visible or the pair (a, b) is visible.

We define a cut as a line segment connecting two visible points in $\mathbf{V}_{t+b}$. Let $p = (x_p, y_p), q = (x_q, y_q)$ be two

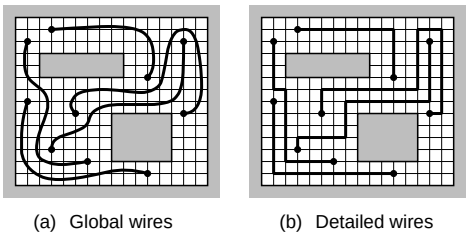

Fig. 1. Routing model with rectangular modules.

terminals of a cut $\overline{pq}$. The capacity $cap(p,q)$ of the cut $\overline{pq}$ is the maximum number of wires which can pass across the cut $\overline{pq}$, that is, $cap(p,q) = \max(|x_p - x_q|, |y_p - y_q|) - 1$. Given the global routing, we define the flow of cut $\overline{pq}$, denoted by $flow(p,q)$, as the number of the wires which pass across the cut $\overline{pq}$. The wire which do not cross $\overline{pq}$ topologically does not contribute to the flow. The wire running from p to q contributes -1 to the flow . We call the constraint that $cap(p,q) \geq flow(p,q)$ the capacity constraint for cut $\overline{pq}$.

The following theorem was given by Cole and Siegel[1].

Theorem 1 Given global routing, the global wires can be transformed into detailed ones if and only if the capacity constraint is satisfied for every cut.

Let l_a^+ (l_a^-) be the line passing through a point a in the routing area with angle 45(-45)-degrees. Let b be a point not on l_a^+ or l_a^-. We denote by $rect(a,b)$ the rectangle formed by the four lines l_a^+, l_a^-, l_b^+ and l_b^-.

Consider a point a=(x_a, y_a) and two lines l_a^+,l_a^-. We define an *upper* area of a as the intersection of the upper areas of the two lines. In the same manner, we define *left, right* and *lower* areas of a. When we rotate the coordinate axes to (-45)-degrees, the new coordinates of a is $(\mu_a, \nu_a) = (\frac{x_a - y_a}{\sqrt{2}}, \frac{x_a + y_a}{\sqrt{2}})$. For a=(μ_a, ν_a) and b=(μ_b, ν_b), if $\mu_a > \mu_b$, we say that a is larger than b with respect to the (-45)-degrees axis. Similary, if $\nu_a > \nu_b$, a is larger than b with respect to the 45-degrees axis.

III. The Capacity Checking Graph

According to Theorem 1, the routability checking can be done by testing the capacity constraint for every cut. In practice, it is enough to check some particular cuts. For instance, consider a triangle abc which does not include a point in $\mathbf{V}_{t+b}$ in its interior or on the boundaries. Suppose that the capacity for cut(a,b) is greater than the sum of the capacities of cut(a,c) and (b,c). Then, even if wires entering the triangle across (a,c) or (b,c) leave it across (a,b), violation of the capacity constraint does not occur. That is, if the capacity constraint for (a,c) and (b,c) are satisfied, then the checking for (a,b) is redundant. In this section, we define the capacity checking graph for representing all the non redundant cuts.

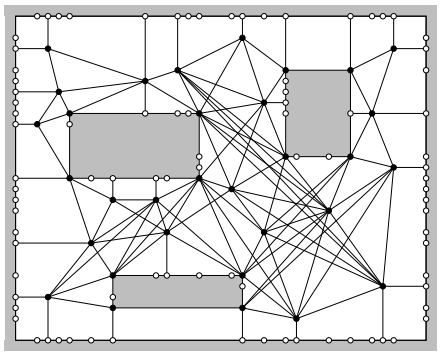

Fig. 2. A capacity checking graph.

A. Definition of the CCG

Let a be a terminal or a module point. A projection point of a is a boundary point which is visible from a and is at the intersection of the horizontal or vertical line passing through a and the boundary of an obstacle. Let $\mathbf{V}_p$ be the set of projection points. A terminal has at most four projection points, and a module point has at most two projection points.

For $a \in \mathbf{V}_{t+b}$ and $b \in \mathbf{V}_{b-m}$, $tri(a,b)$ is defined to be the triangle area surrounded by 45 and (-45)-degrees lines passing through a and a line which supports the boundary containing b. When b is one of the four corner points of the routing area, $tri(a,b)$ is not unique. We can select any of them.

We define a visibility graph $\mathbf{G}_v = (\mathbf{V}_{t+b}, \mathbf{E}_v)$ as follows. $\mathbf{E}_v$ is the set of visible pairs of points in $\mathbf{V}_{t+b}$. According to Theorem 1, the global wires can be transformed into detailed ones if and only if the capacity constraint for every edge of $\mathbf{G}_v$ is satisfied .

The capacity checking graph $\mathbf{G}_c = (\mathbf{V}_c, \mathbf{E}_c)$ is a subgraph of $\mathbf{G}_v$, where $\mathbf{V}_c$ is the union of $\mathbf{V}_{t+m}$ and $\mathbf{V}_p$. $\mathbf{E}_c$ is defined as follows. (i)For $a, b \in \mathbf{V}_{t+m}$, if there is no point $c(\in \mathbf{V}_{t+m})$ visible from a or b in $rect(a,b)$, then $(a,b) \in \mathbf{E}_c$. (ii)For $a \in \mathbf{V}_{t+m}$ and $b \in \mathbf{V}_p$, if b is the projection point of a and there are no points $c(\in \mathbf{V}_{t+m})$ in $tri(a,b)$, then $(a,b) \in \mathbf{E}_c$. (iii)For the remaining cases, (a,b) is not included in $\mathbf{E}_c$.

Figure 2 illustrates an example of the CCG. We have the following theorem.

Theorem 2 Given the global wires, if $cap(a,b) \geq flow(a,b)$ for every edge (a,b) of $\mathbf{G}_c$, then $cap(p,q) \geq flow(p,q)$ for every edge (p,q) of $\mathbf{G}_v$.

According to Theorem 2, if the capacity constraint for the edges of the CCG is tested, then we can decide the routability of the wires.

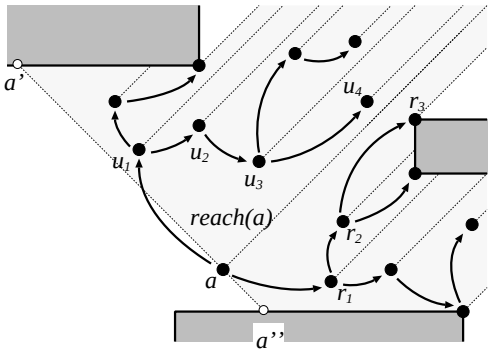

Fig. 3. $Reach(a)$(dotted area) and $\mathbf{Tree}(a)$.

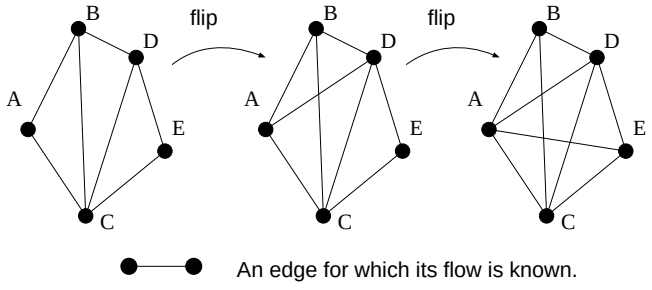

Fig. 4. Flippings. The first flip for $\Box ABDC$ gives $flow(A,D)$. The next flip for $\Box ADEC$ gives $flow(A,E)$.

B. Constructing the CCG

For the routing model which does not include modules, it was shown that the CCG has can be constructed in $O(n \log n)$ time if n terminals are placed randomly in the routing area[3]. In the rest of this section, we present an efficient algorithm for constructing the CCG for the model which includes modules.

The algorithm consists of three steps. (Step 1)We find $\mathbf{V}_p$, that is, the set of projection points of the points in $\mathbf{V}_{t+m}$. (Step 2) We find the edges of the CCG each having both endpoints in $\mathbf{V}_{t+m}$. (Step 3)We find the edges of the CCG each having one endpoint in $\mathbf{V}_{t+m}$ and the other in $\mathbf{V}_p$. Step 1 is done by the sweeping method and it takes $O(|\mathbf{V}_{t+m}| \log |\mathbf{V}_{t+m}|)$ time. Next, we explain Step 2. Let $a (\in \mathbf{V}_{t+m})$ be a point in the routing area. we draw a (-45)-degrees line (a scan line) passing through a. Let a' , a'' be the farthest points on the line visible from a. We draw upper right(45-degrees) half-lines in the upper area of the scan line which start at every module point. We denote by $reach(a)$ the area in the upper side of the scan line where we can reach from a without crossing a module boundary or a half-line starting at a module point(See Figure 3).

Lemma 1 Let a, b be points in the routing area. We assume that they are visible each other and $\nu_a \leq \nu_b$. If $(a,b) \in \mathbf{E}_c$, then b is in $reach(a)$.

According to Lemma 1, it is enough for us to search in $reach(a)$ for finding the edges of $\mathbf{E}_c$ which are incident to a and exist in the upper or right area of a. (We call these edges the upper right edges of a.) We define a reach tree of a and denote it by $\mathbf{Tree}(a)$. Its root is a and every point$(\in \mathbf{V}_{t+m})$ in $reach(a)$ is its vertex. A vertex of the tree has at most two children. One child is called an upper child and the other a right child. An upper child and its descendants exist in the upper area of its parent. A right child and its descendants exist in the right area of its parent(See figure 3). $\mathbf{Tree}(a)$ is called the heap search tree in Computational Geometry.

Lemma 2 Let u_1, r_1 be the upper and right child of a, respectively. Let u_{i+1} be a right child of u_i and r_{j+1} be an upper child of r_j. For each $r_i(u_i)$, the edge connecting a with $r_i(u_i)$ is included in $\mathbf{G}_c$. For other vertices p of $\mathbf{Tree}(a)$, (a,p) is not included in $\mathbf{G}_c$.

According to Lemma 2, we can find the upper right edges of a by searching in $\mathbf{Tree}(a)$. The search time is bounded by the number of the edges of the CCG.

The implementation for Step 2 uses the sweep technique. We draw a (-45)-degrees sweep line in the routing area and scan it from the upper right corner to the lower left corner. The event points are points of $\mathbf{V}_{t+m}$. When the sweep line stops at the event point a, we construct $\mathbf{Tree}(a)$ based on $\mathbf{Tree}$'s which have already constructed. The construction time is proportional to the number of the upper right edges of a in $\mathbf{G}_c$. Then we search in $\mathbf{Tree}(a)$ and output the upper right edges of a. If a is a module point, then we will update the $\mathbf{Tree}$'s. We omit the details.

The total running time for Step 2 is $O(|\mathbf{V}_{t+m}| \log |\mathbf{V}_{t+m}| + |\mathbf{E}_c|)$. And, it is easy to see that Step 3 can be done in $O(|\mathbf{E}_c|)$ time. Therefore, we establish the next theorem.

Theorem 3 The capacity checking graph $\mathbf{G}_c = (\mathbf{V}_c, \mathbf{E}_c)$ can be constructed in $O(n \log n + |\mathbf{E}_c|)$ time, where n is the number of terminals and modules.

IV. The Initial Flow Graph

Lemma 3 Let $\Box v_1 v_2 v_3 v_4 (v_i \in \mathbf{V}_{t+b})$ be a convex rectangle which has no point$(\mathbf{V}_{t+b})$ in its interior. Then, $flow(v_1, v_3) + flow(v_2, v_4) = max(flow(v_1, v_2) + flow(v_3, v_4), flow(v_2, v_3) + flow(v_4, v_1))$.

The proof of Lemma 3 depends on the topologies of the wires passing across the edges of the convex rectangle and is omitted in this version.

According to Lemma 3, given the flows of four edges and one diagonal edge of a convex rectangle, we can compute

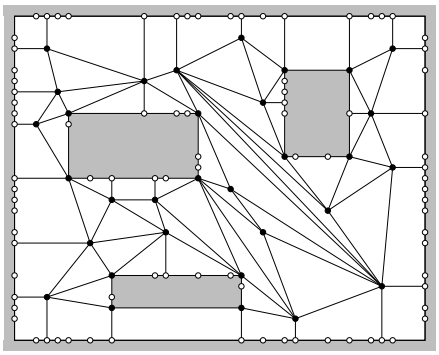

Fig. 5. The initial flow graph.

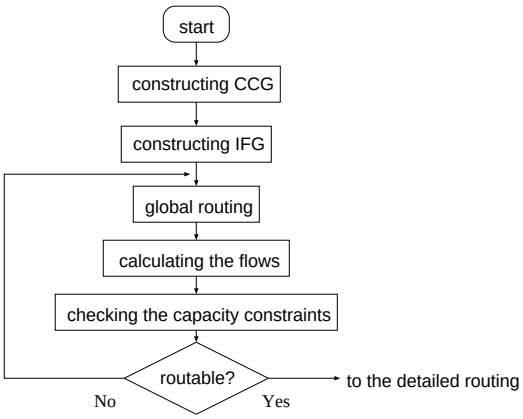

Fig. 6. Flowchart for the routing process.

the flow of the other diagonal edge. We call this operation *flipping*. Figure 4 illustrates an example of the flippings.

The initial flow graph(IFG) is a subgraph $\mathbf{G}_f = (\mathbf{V}_c, \mathbf{E}_f)$ of the CCG. $\mathbf{E}_f$ is defined as follows. We assume that $\nu_a \leq \nu_b$ for points a and b. (i)For $a, b \in \mathbf{V}_{t+m}$, if there is no visible point($\in \mathbf{V}_{t+m}$) from a or b in the intersection of the upper area of a and the left area of b, or in the intersection of the right area of a and the lower area of b, then (a, b) is in $\mathbf{E}_f$. (ii)For $a \in \mathbf{V}_{t+m}$ and $b \in \mathbf{V}_p$, if b is a projected point of a, then (a, b) is included in $\mathbf{G}_f$.

Figure 5 illustrates the IFG. Given a CCG, its IFG $\mathbf{G}_f = (\mathbf{V}_c, \mathbf{E}_f)$ can be constructed in $O(|\mathbf{E}_f|)$ time. It is easy to see that $\mathbf{G}_f$ is planar and the number of the edges $|\mathbf{E}_f|$ is $O(|\mathbf{V}_c|)$. If the flows are given to all edges of $\mathbf{G}_f$, then we can compute the flows of all edges of $\mathbf{G}_c$ using flippings. The computation time is $O(|\mathbf{E}_c|)$.

V. Global routing

Figure 6 illustrates a flowchart of the global routing process using the CCG and the IFG proposed in this paper. It takes $O(n \log n + |\mathbf{E}_c|)$ to construct the CCG and $O(n)$ time to construct the IFG, where n is the number of terminals and modules. Then we do global routing and find the flows for all edges of the IFG. Next, we find the flows of the edges of the CCG and check the capacity constraints in $O(|\mathbf{E}_c|)$ time. If the terminals are placed randomly, then $|\mathbf{E}_c|$ is expected to be $O(n \log n)$ and all operations above are done in $O(n \log n)$ time. If a wire is found to be not routable, we modify some global wires and check the capacity constraints for the involved cuts.

In this routing process, the global routing needs the flows for the edges of the IFG. We can triangulate the routing area using the edges of the IFG. Some existing global routing methods divide the routing area into the Delaunay triangulation and find the global wires. Repeating flippings to the Delaunay trianglation, we can transform it to the triangulation constructed from the IFG. Thus, we find the flows for the edges of the IFG.

VI. Conclusion

We have proposed two graphs, the capacity checking graph and the initial flow graph for efficient routability checking.

The model in this paper assumes a one-layer routing. In the case of the multilayer routing, the planar global routing is done on each layer, after the assignment of the nets to layers. When wires of other layers are to be connected, we use through-holes which we can treat as terminals in the routing area.

Acknowledgements

The authors wish to thank Makoto Ito of Chukyo University and Xuehou Tan of Tokai University for their helpful comments.

References

[1] R.Cole and A.Siegel : "River routing every which way, but loose," *Proceeding of 25th annual Symposium on Foundations of Computer Science*, pp. 65–73, 1984.

[2] W.W.Dai, R.Kong and M.Sato : "Routability of a rubber-band sketch," *28th ACM/IEEE Design Automation Conference*, pp. 45–48, 1991.

[3] Y.Kawaguchi, N.Iso and T.Hirata : "Two graphs for efficient routability checking," *The Trans. IEICE*, Vol. J79–11, to appear, in Japanese, 1996.

[4] C.E.Leiserson and F.M.Maley : "Algorithms for routing and testing routability of planar VLSI layouts," *Proceeding of the 17th Annual ACM Symposium on Theory of Computing*, pp. 69–78, 1985.