Bottlenecks Identification in General Junctions*

Ahsan Abdullah
Department of Computer Science
University of Southern California
Los Angeles, CA 90089, USA

Abstract

Consider n non-overlapping rectangular polygons. Let the interpolygon space be denoted by ρ, ρ is rectangular if all polygons are identical. However, for variable size rectangular polygons, ρ is non-rectangular with rectilinear boundary, let us denoted this by $\overline{\rho}$. $\overline{\rho}$ can be divided into rectangles, and the union of these rectangles is called a *junction*. $\overline{\rho}$ may have L-, S-, T- and X-shaped junctions. We consider the problem of identifying bottlenecks in general junctions. We present an algorithm, which uses a simple data structure, and identifies the bottlenecks in $O(n)$ time and space. Furthermore, the algorithm can also be used to generate the visibility graph for rectilinear visibilities.

1 Introduction

The building block strategy for VLSI layout consists of two sub-problems, *Placement* and *Routing*. *Placement* consists of assigning positions to modules (e.g. ICs, Macro cells) on a carrier (e.g. Printed Circuit Board, Silicon Wafer) according to some objective function, while *Routing* consists of assigning paths to interconnections among modules through the intermodule space (channel). When variable size modules are used, non rectangular channels with rectilinear sides are created. The channel space can then be divided into rectangles, and the union of these rectangles is called a *junction*. For Building Block layout, L-, S-, T- and X- shaped junctions are created. In this paper, we consider the problem of identifying the routing bottlenecks in the channels of general junctions.

L-, T- and X- shaped non-skewed junctions were first considered in [13]. In [10, 11] skewed junctions were also considered, and bounds were given for the widths of the channels of general junctions. The bounds were dependent on the crossings of interconnect in the junctions.

One solution to bottleneck identification could be based on "simple" visibility check. Two modules are said to be visible, if they can be joined by a line, such that the line does not intersects with any other module/s, this is repeated for all pairs of visible modules. The bottlenecks could then be identified as the minimum length visibility lines. However, this approach may not be suitable due to the following reasons: i) The fastest algorithm for generating the visibility graph for disjoint polygons takes time $O(E + n \log n)$ [5], here E could be $O(n^2)$. This problem was also solved by [2, 9] and [16] in time $O(n^2 \log n)$, $O(n^2)$ and $O(n^2)$ respectively. ii) Even if we had a faster algorithm, it is not necessary to generate all the visibility lines. Consider Fig. 2(f), lines are drawn between all pairs of visible modules. Fig. 2(c) shows the lines generated by our algorithm. Note that the lines are fewer, asymptotically they could be $O(n)$ times fewer.

We make the following assumptions i) Rectangular modules are placed on an invisible rectangular grid, such that there are no modules smaller than the unit square (of the grid). ii) The dimensions of the modules i.e. Height (H) and Width (W) are $O(1)$ w.r.t to the number of modules. iii) The distance between modules is $O(1)$. Note that these assumptions are not impractical. Since, asymptotically, the dimensions of the modules can be considered to be constant. Also, the length of the interconnect is proportional to the distance between modules, therefore, closely packed modules reduce that length.

2 Definitions and Notation

The *junction* will be defined as in [11, 10]. An L-*junction* $\mathrm{L}(t_1, t_2, x_o, y_o)$ is an "L" shaped region, which is the union of the following rectangular regions (see Fig. 1(a)):

$L = \{(x, y) : x < 0, -t_1 \leq y \leq 0\}$
$T = \{(x, y) : 0 \leq x \leq t_2, 0 < y\}$
$J_1 = \{(x, y) : 0 \leq x \leq x_o, -t_1 \leq y \leq 0\}$
$J_2 = \{(x, y) : 0 \leq x \leq t_2, -y_o \leq y \leq 0\}$

The sets L and T are referred to as the *left* and *top* channels and $J_1 \cap J_2$ as the *junction region*. Any shortest Manhattan path between (0,0) and $(x_o, -y_o)$ is called the *bottleneck* of the junction region. A non-skewed L-junction corresponds to $x_o = t_2$ and $y_o = t_1$. The S-, T- and X- junctions are shown in Fig. 1(b)-(d), and can be defined similar to L-*junction*, see [10]. Fig. 2(e) shows some junctions, e.g M_1, M_2, M_5 and M_7 define an L-junction, M_5, M_7, M_{12} and M_{11} define a T-junction, and M_5, M_4, M_9 and M_{11} define an X-junction.

*This research was supported by S & T Scholarship, Govt. of Pakistan.

Consider module M_i, let (X_{Li}, Y_{Bi}) be the bottom left vertex of M_i, and let (X_{Ri}, Y_{Ti}) be the top right vertex. For a module M_i, only these two vertices are stored. The i^{th} vertical line between the top and bottom of the carrier [1] is denoted by V_i, H_i is defined similarly. H_o is the top edge of the carrier and V_o it's left edge. V_{last} is the last V line put back, after it was tentatively removed in procedure **Line-Removal**, initially $V_{last} = V_o$. The minimum length visibility line between a vertex and an edge of modules M_i and M_j is denoted by $D_{(i,j)}$

SV_i is the set of all modules which intersect with V_i, $SV_{(i,j)} = SV_i \cup SV_j$, similar definitions for SH_j and $SH_{(i,j)}$. $SVH_{(i,j)}$=$SV_{(last,i+1)} \cap SH_{(j,j+1)}$. When V_i is to be removed in procedure **Line Removal**, a rectangle, WINDOW is defined by V_{last}, V_{i+1} and two adjacent H lines. A CELL is a rectangle defined by V_i, V_{i+1} and H_j, H_{j+1}, it is used in procedure **Neighbor-Check**, GRID[i+1, j+1]=CELL. $\alpha \in (-1, 0, +1), \beta \in (-1, +1)$, e.g $i + \alpha$ could be i-1, i, i+1, similar result for $i + \beta$. Note that GRID$[i + \alpha, j + \alpha]$ is invalid for $\alpha = 0$.

3 Procedure descriptions

A brief description of four main procedures used in our algorithm.

3.1 Line-Generation

As mentioned in **Introduction**, we assume all modules to be on an invisible rectangular grid. The purpose of this procedure, is to make only those grid lines "visible", which intersect with (X_{Li}, Y_{Bi}) and (X_{Ri}, Y_{Ti}). Furthermore, the procedure also identifies those modules, whose vertices do not intersect with any of the V/H grid lines, but their edges do intersect. In Fig. 2(b), for V grid lines, top and bottom edges of M_5 intersect with V_3 and V_4.

3.2 Line-Removal

This procedure will discretize the carrier. This is achieved by selectively removing only those grid lines, such that, atmost one module, or part of one module lies inside the WINDOW. Consider modules $M_k, M_l \in SVH_{(i,j)}$, if any vertex of both modules lies inside the WINDOW, then V_i is not removed and $V_{last} = V_i$. Note that there can be a constant number of vertices inside a WINDOW. Techniques are given in [14] to determine a point inside a rectangle. In Fig. 2(b) dotted lines are removed.

3.3 Neighbor-Check

After **Line-Removal**, each CELL will have atmost a single, or part of a single module. This enables **Neighbor-Check** to "know" the location of the modules w.r.t each other. Furthermore, there is no need to check the visibility between say GRID[1,1] and GRID[5,5]. Since, they may not be visible from each other because of the modules in the other 11 GRID[ ]s obstructing the visibility line. However, even if a visibility line was possible, it would be eliminated in the **Min-Distance procedure**, therefore, there is no need to compute it in the first place. Hence, it is only necessary to note the minimum length visibility lines between GRID[i, j] and GRID$[i + \alpha, j + \alpha]$.

$D_{(x,y)}$ is generated between GRID[i, j]($\neq \phi$) and:
1. GRID[i+α, j+α] IF GRID[i+α, j+α] $\neq \phi$.
2. GRID$[e_i + \alpha, e_j + \alpha] \neq \phi$ for every GRID[i+β, j+β]=GRID$[e_i, e_j]$=ϕ.
3. Every GRID$[h_i + h, j + \alpha] \neq \phi$ IF atleast one GRID[i+β, j]=GRID$[h_i, j]$=ϕ. Note that h is some possitive constant. Similar result for GRID[i, j+β].

This procedure will also store the end points of $D_{(x,y)}$ at GRID[i, j] for M_x, at GRID[a,b] for M_y, and in the GRID[ ]s defined by the Grid Vector (GV), having GRID[i, j] and GRID[a,b] as the two end "points".

3.4 Min-Distance

This procedure decides which $D_{(i,j)}$ to keep for the pairs of visible modules identified. Consider two intersecting lines $D_{(a,b)}$ and $D_{(x,y)}$ among k [2] lines whose end points are stored at GRID[i, j]. Also assume that, they are neither vertical nor horizontal. Let $R_{(a,b)}$ and $R_{(x,y)}$ be two rectangles having $D_{(a,b)}$ and $D_{(x,y)}$ as their diagonals, respectively. Only that D will be kept, for which no module M in GRID[a+α, b+α] and GRID[x+α, y+α] intersects with the rectangle R corresponding to D. For vertical and horizontal intersecting lines inside GRID[i, j], shortest is kept. If both lines equal in length, either V or H is kept. Fig. 2(c) shows the necessary visibility lines generated, and Fig. 2(d) shows the corresponding bottlenecks.

For any two modules in neighboring GRID[ ]s, we generate all sixteen lines between them and then pick the shortest visible line. Since number of intermodule visibility lines is constant, therefore, we do not use efficient techniques such as given in [4, 12, 15].

4 Computational Complexity

As a result of assumptions ii) and iii) in **Introduction**, the area of the rectangular carrier will be $O(n)$. There can be two extremes, such as, W=$O(n)$, H= $O(1)$, and W=$O(1)$, H=$O(n)$, and in-between we can have other values of H and L, such as W=H=$O(n^{0.5})$. Furthermore, the variation in the positions of the modules will be $O(n)$, therefore, we can use bin-sort to sort X_{Li}, X_{Ri} and Y_{Ti}, Y_{Bi} so as to assign M_i to V_j and H_k.

S_V, S_H and V, H lines are implemented by sorted linked lists. During different operations, the lists are kept sorted by using pointers to the modules being considered. To get $SVH_{(i,j)}$ we use the technique in [1] with selective checking, therefore, $SVH_{(i,j)}$ takes time $O(1)$. The complexity of different procedures depends

[1] Assuming a rectangular carrier.

[2] $0 \leq k \leq \delta, \delta$ is some constant.

on the size of the grid, which is $O(n)$. Therefore, the complexity of our algorithm is $O(n)$. Note that, Line-Removal has to be used for removing H lines also.

References

[1] Aho, Hopcroft and Ullman *Data Structures and Algorithms*, Addison-Wesley, pp. 115-117, 1983.

[2] T. Asano, T. Asano, L. Guibas, J. Hershberger and H. Imai, "Visibility of disjoint polygons," *Algorithmica*, 1 (1986), pp. 49-63.

[3] B. Chazelle "Triangulating a simple polygon in Linear time" 1990 *IEEE 31st Symp. on Foundations of Computer Science*, pp. 220-229.

[4] Chin, and Wang "Optimal Algorithms for the Intersection and the Minimum Distance problems Between Planar Polygons," *IEEE Trans. on Computers*, vol. c-32, no. 12, pp. 1203-1207, Dec. 1983.

[5] S. K. Ghosh, and D. M. Mount "An ouput sensitive algorithm for computing visibility graphs" *Proc. 28th IEEE Syp. on Foundations of Computer Science*, pp. 11-19, 1987.

[6] L. Guibas, J. Hershbereger, D. Leven, M. Sharir and R. E. Tarjan "Linear time algorithms for visibility and shortest path problems inside triangulated simple polygons," *Algorithmica* 2(1987) pp. 209-233.

[7] J. Hershberger, "An optimal Visibility graph Algorithm for triangulated simple polygons," *Algorithmica* 4(1989), pp. 141-155.

[8] D. G. Kirkpatrick, M. M. Klawe, R. E. Tarjan, "$O(n \log \log n)$ polygon triangulation with simple datastructures," *Proc. 6th ACM Symp. on Computational Geometry*, 1990, pp. 34-43.

[9] D. T. Lee, "Proximity and reachability in the plane," Ph.D Thesis, Tech. Rep. ACT-12, Coordinated Science Lab., Univ. of IL, 1978.

[10] S. R. Middila and D. Zhou "Lower and upper bounds for the general junction routing problem," Tech. Rep. ACT-90, Coordinated Science Lab., Univ. of IL, May 1988.

[11] S. R. Middila and D. Zhou "Routing in general junctions," *IEEE Trans. on CAD*, vol. 8, no. 11, Nov. 1989, pp. 1174-1184.

[12] M. McKenna and G. T. Toussaint "Finding the minimum vertex distance between two disjoint convex modules in linear time," Report No. SOCS-83.6, School of Computer Science McGill Univ., Montreal, Quebec 1983.

[13] P. Y. Pinter, "Optimal routing in rectilinear channels," in *VLSI Systems and computations*, H. T. Kung, B. Sproull, and G. Steel, (eds.), pp. 160-177, 1981.

[14] F. P. Preparata and M. I. Shamos *Computational Geometry an Introduction*, Springer-Verlag, 1985.

[15] J. T. Schwartz, "Finding the minimum distance between two convex polygons," *Inform. Process. Lett.*, vol. 13, no.4 and 5, pp. 168-170, 1981.

[16] E. Welzl, "Constructing the visibility graph for n line segments in $O(n^2)$ time," *Inf. Proc. Lett.*, pp. 167-171., 20(1985).

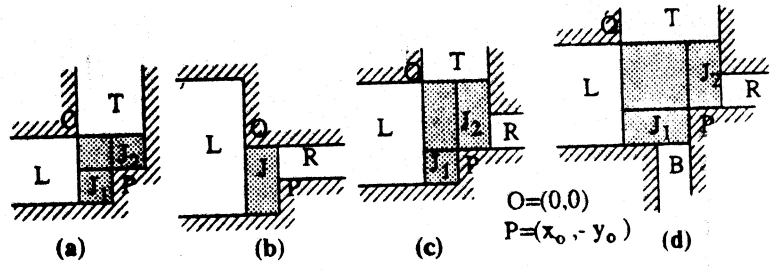

Figure 1: L-, S-, T- and X-junctions.

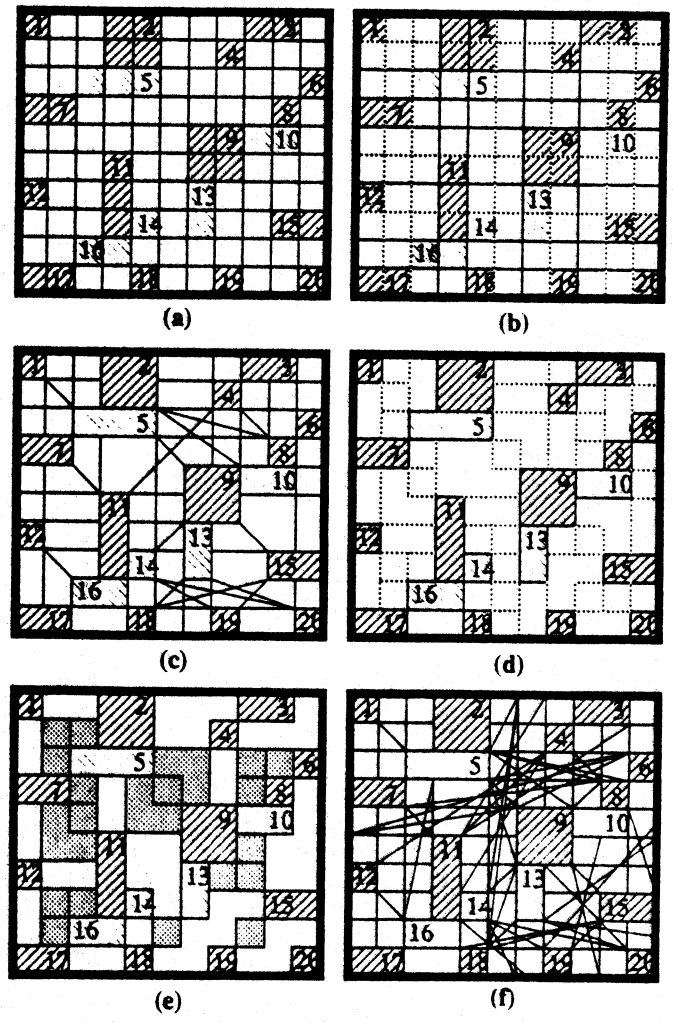

Figure 2: Junction bottleneck identification.