Topological Routing in SURF: Generating a Rubber-Band Sketch

Wayne Wei-Ming Dai Tal Dayan David Staepelaere

Computer Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064 USA

ABSTRACT

A multi-layer topological router for generating rubber-band sketches is described. The router uses hierarchical top-down partitioning to perform global routing for all nets simultaneously. It combines this with successive refinement to help correct mistakes made before more detailed local information is discovered. Layer assignment is performed during the partitioning process to generate routing that has fewer vias and is not restricted to one-layer one-direction. The local router uses a region connectivity graph to generate shortest-path rubber-band routing.

1 Introduction

This paper describes the topological router at the heart of the SURF routing system. This router is designed primarily for routing MCM substrates, which consist of multiple layers of free (channelless) wiring space. Because MCM substrate designs have potentially large numbers of terminals and nets, the router must be able to handle large designs efficiently in both time and space. In addition, the router should be flexible and permit an incremental design process. That is, when small changes are made to the design, it should be able to be updated incrementally and not recreated "from scratch". This allows quicker convergence to a final design. In order to produce designs with fewer vias, the router should be able to relax the one-layer one-direction restriction. This is an important consideration in high speed designs since the discontinuities in the wiring caused by bends and vias are a limiting factor for system clock speed.

In order to support the flexibility described above, the router must have an underlying data representation that models planar wiring in a way that can be updated locally and incrementally. For this reason, SURF models wiring as rubber-bands [3,8]. Rubber bands provide a canonical representation for planar topological wiring. Because rubber bands can be stretched or bent around objects, this representation permits incremental changes to be made that only affect a local portion of the design. For a discussion of how this representation can be maintained and updated dynamically, see [4]. Once the topology of the wiring is known, the rubber-band sketch can be augmented with *spokes* to express spatial design constraints such as wire width, wire spacing, via size, etc. [5]. Since successful creation of the *spoke sketch* guarantees the existence of a geometrical wiring (Manhattan or octilinear), the final transformation to fixed geometry wiring can be delayed until later in the design process. This allows most of the manipulation to take place in the more flexible rubber-band format. Figure 1 shows several different views of the same wiring topology. These represent various states of the rubber-band representation.

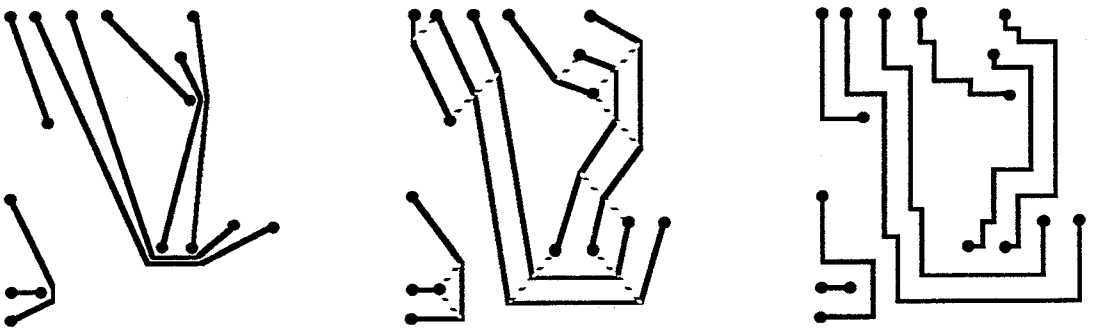
The subject of this paper is a topological router that produces multi-layer rubber-band sketches. The input to this router is a set of terminals, a set of nets, a set of obstacles, and a set of wiring rules. These rules include geometrical design rules and constraints on the wiring topology. The topological constraints may include valid topologies (daisy chain, star, etc.) as well as absolute and relative bounds on segment lengths. The output of the router is a multi-layer rubber-band sketch in which all the points of a given net are connected by wiring.

Although the routability of a sketch is not guaranteed until the successful creation of spokes, at each stage, the router uses the increasingly detailed information available to generate a sketch without overflow regions. This increases the chance that the sketch can be successfully transformed into a representation (the spoke sketch) that satisfies all of the spatial constraints. In addition the router tries to reduce overall wire length and the number of vias.

2 Overview of our Approach

The topological routing is done in two steps: global routing and local routing. The global router determines rough net topology and partitions the routing area into a set of bins. The interfaces between individual bins are specified by placing crossing points on the bin boundaries for each net that crosses the boundary. These cross points specify a crossing position and layer. The local router then routes the individual bins independently.

Our global routing approach employs two principles from the field of Artificial Intelligence—*the least commitment principle* and the notion of *maximal use of information*. The least commitment principle states that if the correct choice in a decision is not known, the decision should be delayed. This guards against making arbitrary decisions early that, if wrong, could adversely affect the outcome. Maximal use of information states that all available, relevant information should be applied to solving the problem. In early stages of the partitioning, we reduce our commitment by allowing the cross points a wide range of movement along the cut line. Later, after further partitioning, we use the new and more detailed information to reassign the cross points. The only commitment at each level is the assignment of cross points to specific interfaces. When a cross point is assigned to an interface, it will remain there.



(a) Rubber-band sketch (b) Extended rubber-band sketch (c) Geometrical wiring

Figure 1.1: Views of a rubber-band sketch. The rubber-band sketch (a) is a canonical representation of planar topological routing in which wires are treated as elastic rubber-bands. Figure (b) shows the extended rubber-band sketch (spoke sketch). In this sketch, spokes are used to separate wires and satisfy spatial constraints. Successful generation of (b) guarantees that the transformation to geometrical wiring (c) is possible.

Once the global router has partitioned the problem into bins, the local routing is performed. The local routing is done one net at a time within the limits of a bin. This limits the search to a single bin and improves the time and space efficiency of the router. The local router uses a *region graph* which represents the geometrical and topological adjacency of areas of the sketch. By using this graph, the router can efficiently find the shortest planar path through the partially routed bin.

Both the global and the local router rely heavily on an underlying data structure built on constrained Delaunay triangulation [2]. The Delaunay triangulation of a set of points is the straight-line dual of the Voronoi diagram for that set. An important property of the Delaunay triangulation is that the circumcircle of each triangle contains no points in its interior. Based on Delaunay triangulation, such problems as closest pair, all nearest neighbors, and Euclidean minimum spanning tree (MST) in the plane can be solved in linear time. *Constrained* Delaunay triangulation includes edges that are forced to be part of the triangulation. SURF uses constrained edges to represent rubber-band segments, obstacles, etc. The global router relies on fast MST generation for calculating the cost matrices used to determine crossing point locations. The local router relies on the triangulation for performing shortest path calculations within its region graph.

3 Global Routing

The approach taken by SURF for global routing is a hierarchical top-down partitioning technique based on the approaches described in [6], [7] and [9]. The basic approach is to repeatedly divide the routing problem into smaller subproblems (bins) until they reach an appropriate size for the local router. At every step, each of the currently active subproblems is partitioned into two smaller problems. The approach taken to partition a single bin is: determine a cut line that divides the bin into two pieces, and for each net in the bin that crosses the cut line, determine a place along the cut line for the net to cross. New artificial points are generated for each of the crossing points. The cut line and crossing points form an *interface* between the two smaller bins they create. Although the cross points are tentatively assigned positions and layers along the interface, they are not fixed, and may be repositioned and reordered within the limits of their interface.

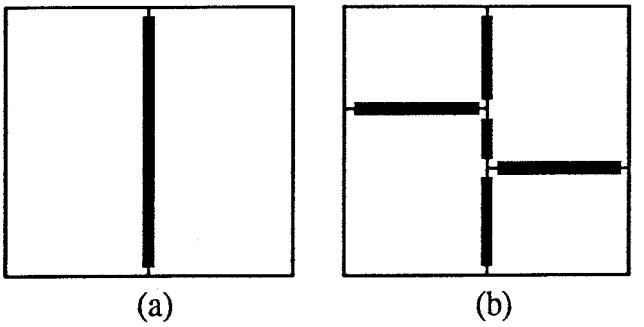


Figure 3.1: Interface partitioning. Figure (a) shows a two bins separated by a bin interface (thick line). In Figure (b) the two bins have been further subdivided. This partitions the original interface into three portions.

As the partitioning process continues, the existing interfaces are divided (see Figure 3). When this happens, the set of cross points along existing cut lines are partitioned among the new set of interfaces. This process increasingly restricts the range of movement of the cross points.

The sketch is divided until all of the subproblems reach an appropriate complexity for the local router. Because the sizes and shapes of bins are flexible, the decision of when to stop cutting is made individually for each bin. The factors used for determining a bin's complexity include: the number of nets in the bin, the bin's area, and the number of crossing points and terminals included in the bin. If a bin is partitioned too far, the problem for the local router may be more restricted than necessary.

file	*max*		*min & max*		*min*	
	bins	crosses	bins	crosses	bins	crosses
r200	37	520	33	435	28	347
r300	82	1196	74	968	54	693
r400	149	2268	141	1915	117	1533
r500	242	3723	208	2978	179	2444
ds15	190	2919	140	1925	119	1565

Table 3.1: This table shows the effect of varying the critical cut threshold for several examples. The *max* column uses a low criticality threshold, the *min* column uses a high threshold, and the *min & max* column uses an intermediate value.

3.1 Cut Line Selection

One of the critical decisions in the partitioning process is selecting the proper cut line. For any given subproblem, the choice has to be made whether to cut vertically or horizontally and where along the x- or y-axis to make the cut. The decision of which cut line to select is made by examining the aspect ratio, the net flow, and the cut capacity.

When dividing bins, it is desirable to keep the bins as close to square as possible. For this reason, the orientation of the cut is chosen so that it cuts the larger axis of the bin.

To select the position of the cut, the flow densities of various cuts are considered. For any cut the *flow* and *capacity* are defined as follows. The flow is the number of nets that must cross the cut. These are nets that have points (terminals or crossing points) on both sides of the cut. The capacity is the the number of net crossings that the cut can accommodate. The flow density is the ratio of flow to capacity. The position of the cut is chosen so that it either minimizes or maximizes the flow density. The choice is based on the value of the maximum flow density. If the maximum flow density is above some threshold, the cut associated with it is considered *critical* and chosen as the cut position. This allows dense regions of problem to be considered earlier in the partitioning process and the flow to be distributed more evenly by the assignment of crossing points along the cut.

If the maximum flow density is below the threshold, the minimum flow cut position is chosen. In this case, the cut line generates fewer crossing points. By using fewer crossing points, the amount of commitment at this stage is reduced and the cut line separates the bin into smaller subproblems that are less dependent on one another. Table 3.1 shows the results of varying the critical cut threshold for several examples. In general, min cuts lead to fewer bins and fewer crossing points.

3.2 Cross Point Assignment

Once a cut line has been selected for a given bin, crossing points (tentative positions and layers) are determined for each net that crosses the cut line. This is accomplished by dividing the cut line into a set of crossing slots. Assigning each net to one of these slots determines rough crossing positions and layers. The proper matching of nets to slots is crucial to the approach because it directly affects the overall routability of the problem as well as such characteristics as wire length and the number of vias.

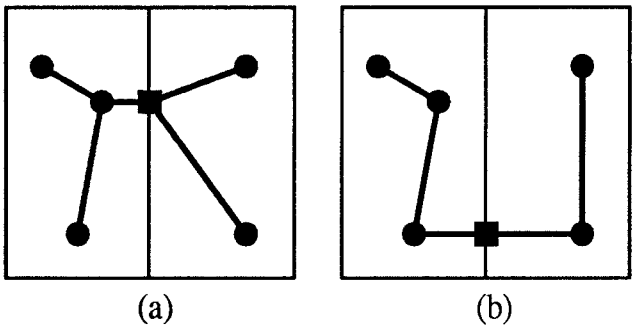


Figure 3.2: Wire estimation using MSTs. Figures (a) and (b) show the MSTs generated to estimate the wiring length cost component for a net using two different crossing point locations.

The matching of nets to slots is done by modeling the problem as an integer network flow problem and determining a max-flow, min-cost solution [7,9]. The problem is formulated as a complete bipartite graph $V = G(V_n, V_s, E)$ where V_n is the set of nets that need to cross the cut line and V_s is the set of crossing slots. Each net $n \in V_n$ represents a unit source. Each crossing slot $s \in V_s$ is a sink with a capacity determined by the size of the slot and the obstacles in the slot. Each edge (n, s) has unit capacity and a cost associated with the desirability of assigning net n to slot s. The assignment of nets to slots is determined by finding a maximum flow assignment with a minimum cost.

The cost of each edge (n, s) is a linear combination of three components: a wire length cost, a via cost, and a routability cost. The wire length component represents the cost due to wire length associated with assigning net n to slot s. This component depends only on the slot's position and not on its layer. The wiring length component for (n, s), is estimated as follows. Two temporary planes are constructed containing the points in the bin that are part of net n. One plane holds the points on one side of the cut; the other plane holds points on the other side of the cut. In each plane, an additional point is added at the position of slot s. An MST is then generated for each of the planes. The total wire length associated with this slot position is then estimated by summing the lengths of the MST edges in both planes. See Figure 3.2. Because the planes are triangulated with Delaunay triangulation, MSTs can be generated efficiently.

The via cost component is used to reduce unnecessary plane switching. It depends on a slot's layer and not its position. For net n, the number of points belonging to each layer on each side of the cut line is counted. If the points on both sides are assigned exclusively to a particular layer, that layer is given a lower cost.

The routability cost is intended to increase the chances of successfully routing the sketch by encouraging wiring that is roughly horizontal to lie in one layer while routing that is generally vertical to lie in the other layer. This is approximated by adding an additional cost to all slots on one layer if the cut is vertical or the other layer if the cut line is horizontal.

3.3 Interface Refinement

As the partitioning process proceeds, the interfaces between bins are repeatedly divided by the cut lines into smaller interfaces. At each step, when an interface is partitioned, the tentative positions of the crossing points in the original interface are used to assign them to one of the new smaller interfaces. At each stage, the additional information gained by the further partitioning is used to refine the positions of the crossing points. This refinement is accomplished by performing another linear assignment pass on the individual interface partitions. Figure 3.3 shows how interface crossing point reassignment can be used to improve the positions of crossing points. The gray regions could represent either obstacles or dense wiring. Before refinement, the crossing point generated in Figure 3.3(a) leads to an unnecessarily long wire in Figure 3.3(b). Figure 3.3(c) shows how the refinement in position leads to a path with shorter length. This technique allows information gained at lower levels of the partitioning process to be used to correct some of the mistakes made at higher levels.

3.4 Backtracking

As the partitioning process continues, it is possible that a mistake made at a higher level will be detected at a lower level. For example, the flow density of a cut is calculated as an average across the entire cut. It is possible that some locally dense region may be overlooked at higher levels of the partitioning. By the time the dense region is discovered, the problem may be too restricted to cut it without producing an overflow cut. Since it is unlikely that a routable solution will be generated for a partitioning with an overflowing cut, the router will attempt to rectify the situation by backtracking and cutting the critical region at a higher level.

The bin set is naturally represented as a binary tree. In this tree, the internal nodes represent cuts and the leaves represent bins. Backtracking to a given level is equivalent to merging some subtree into a single leaf. This merging process can be performed efficiently by removing all of the crossing points contained within the subtree. At this point, a new cut line can be chosen and the process allowed to continue as usual.

4 Local Routing

Once the rough net topology is determined by the global router, the local router generates wiring that conforms to this topology. It routes the bins produced by the global router one at a time. Each bin consists of a rectangular portion of the problem. Points in the bin are either crossing points on the bin boundaries or terminals located within the bin. The goal of the local router is, for each net in the bin, to connect all of its points by legal multi-layer rubber-band wiring. In addition, in order to increase the quality of the results, the router should avoid creating local over-congestion.

4.1 Approach

The basic approach used by the local router is to route the nets one at a time using a shortest path algorithm. Because bins represent small portions of the entire routing area, the number of nets within a bin is limited and routing them one at a time is not a serious drawback.

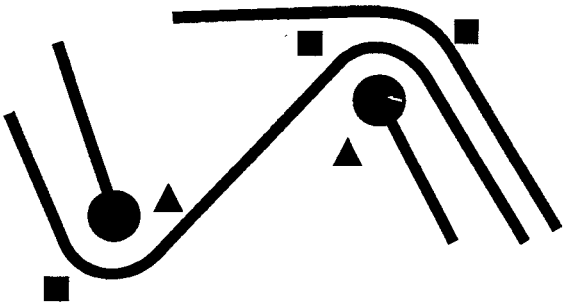


Figure 4.1: Regions of two rubber-band points. The triangles represent regions incident to the rubber-band point. Squares represent attached regions. Two regions are adjacent if they can be connected with a wire that does not cross existing wires.

The strategy is to transform partial nets into complete nets by adding points to them one at a time. Adding a point to an existing partial net consists of two steps: (1) finding a topological path from the point to the existing partial net, and (2) transforming the topological path into a valid rubber-band path.

Finding a Path

The first step when adding a point to an existing partial net is to find a topological path from the point to the partial net. In order to find such paths, a *region graph* is used to represent topological adjacency information. The region graph is derived from the rubber-band representation of the bin. The nodes in this graph are *regions*. Regions represent the connected topological areas in the infinitesimally small neighborhood of a rubber-band point (terminal, via, bend point). Figure 4.1 shows the regions associated with two rubber-band points. The triangles denote *incident* regions. Incident regions are those which actually "touch" the point in question. The squares denote *attached* regions. Attached regions are formed by rubber-band wires bending around a point. The path is determined by using Dijkstra's shortest path algorithm to search from the incident region of the new point to any region incident on the partial net.

The edges in the region graph represent information about geometrical and topological adjacency. Two regions are topologically adjacent if they can be connected with planar wiring. Two regions in the region graph share an edge if they are topologically adjacent and their points share an edge in the constrained Delaunay triangulation (CDT). Because edges in the CDT connect points locally, only considering adjacencies along these edges is an efficient way of representing topological adjacency in a linear number of edges. In Figure 4.1 the incident region of the upper point and the attached region of the lower point are adjacent because they share an edge in the triangulation (in this case a constrained rubber-band edge) and may be connected with a wire that does not cross other wires. In the current implementation

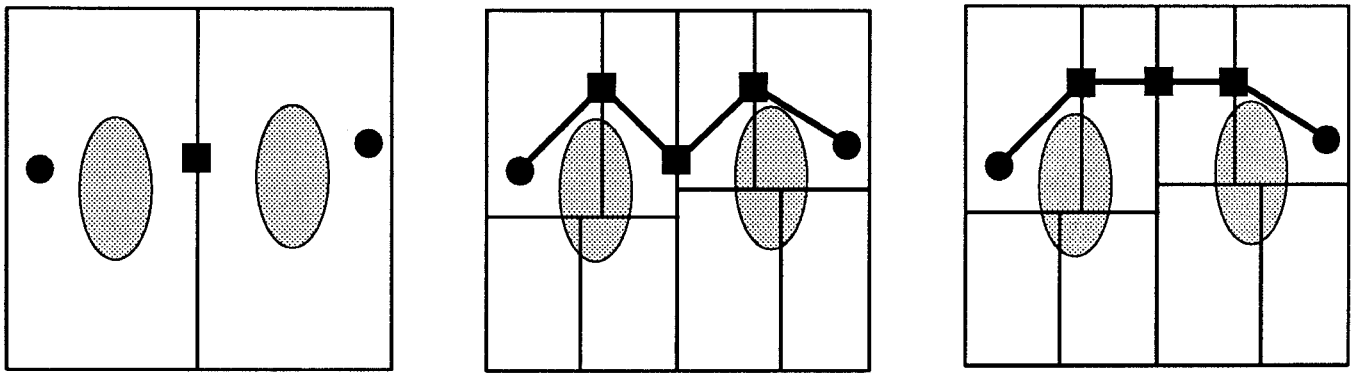


Figure 3.3: Example of interface refinement. In Figure (a) a crossing point is assigned before information about dense regions is discovered. Figure (b) shows the locations of crossing points on further cut lines. These are positioned to avoid the dense regions. In (c) the original crossing point has been refined using the information from (b).

of the local router, the region adjacency edges are not represented explicitly, instead they are derived as needed while traversing the graph.

Validating the Path

Once a path from a new net point to the existing partial net has been found in the region graph, a valid rubber-band path must be created. The path specified by the region search is guaranteed to be planar but is not guaranteed to be a valid rubber-band. For example, the region path shown in Figure 4.1(a) is not a valid rubber-band because the net is attached to point C improperly. The validation process is an iterative one that replaces each pair of invalid segments with a set of zero or more valid ones. Figure 4.1(b) shows the results of validating the wire segments attached to point C.

4.2 Multi-layer Routing

The algorithm described above does topological local routing on a single layer sketch. The extension to multi-layer routing can be summarized as follows. For each point, the new region graph has regions on all of the layers. Two regions are considered adjacent if they are on the same layer and adjacent in the single-layer case, or if they are on different layers and belong to the same point. This extension to the definition of adjacency represents the fact that two regions on different layers can be connected by adding a via in the neighborhood of the point. The cost of edges between regions on adjacent layers represents the cost of adding a via between the two layers.

If the path in the region graph for the multi-layer case includes vias, they are added to the sketch. The paths in each layer are then validated independently as in the single-layer case.

4.3 Avoiding Local Congestion

The path search algorithm described above considers only Euclidean path length and via costs. As a result it may produce areas of local congestion. This may lead to an unroutable sketch. Cole and Siegel present a number of concepts relating to the routability of a rubber-band sketch. A *cut* is a line drawn between two points in a rubber-band sketch. The *flow* of a cut is the number of nets that cross

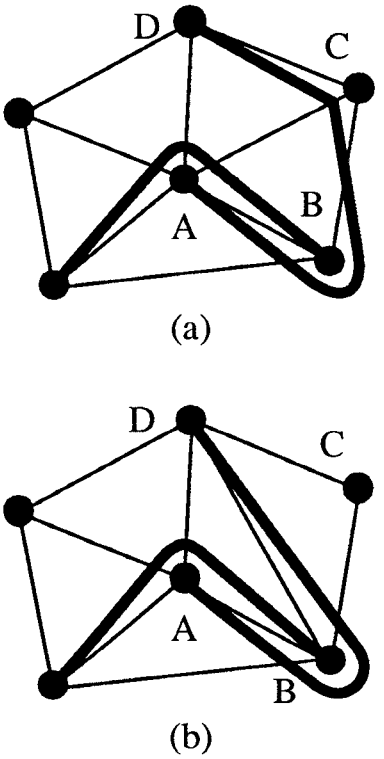


Figure 4.2: Path validation. In Figure (a), the path discovered by a search from A to D has an invalid attachment point at C. Figure (b) shows the path after validation.

the cut. The *capacity* of a cut is the maximum number of nets that may cross the cut without violating spacing constraints. In [3], they prove that a sketch is routable if and only if for all cuts, the flow does not exceed the capacity. The local routing algorithm described above generates valid rubber-band nets but ignores cut capacities. As a result, local congestion may occur. Our approach to avoid such congestion is to satisfy the flow/capacity constraints for the triangle edges in the sketch. This is a good approximation and avoids most of the congestion because the Delaunay triangulation is constructed so that close points share edges. As the flow of a triangle edge reaches capacity, the cost of crossing that edge increases and such paths will be avoided during the shortest path search.

5 Crossing Point Removal

The crossing points created by the global router define the interfaces between the bins. These interfaces allow the local router to operate on bins independently. Once the local routing is completed, the crossing points merely add unnecessary constraints. At this point, if a crossing point is of degree two (one wire connected to it from each bin), it may be removed and the two rubber-band segments incident on it merged into a single rubber-band segment. Normally, the areas around the boundaries are the most difficult portions of a bin to route. This operation will give the rubber-bands more freedom of movement along the bin interfaces and, in general, lead to better results.

6 Conclusion

The rubber-band router described in this paper is based on approaches used previously to solve similar problems. Hu and Shing described the partitioning approach (the *cut and paste* approach) in [6]. Lauther described a hierarchical top-down router for channelless Sea-of-Gates routing [7]. He used linear assignment to assign crossing points to cut lines. Marek-Sadowska described a similar approach for routing macro-cell layouts [9]. Parng and Tsay described additional improvements for Sea-of-Gates routing [10].

The SURF router is designed to tackle a slightly different problem than any of the previous work—generating a topological specification for flexible rubber-band wiring. This system is intended for routing across entire MCM substrates and has no notion of channels. Routing for Sea-of-Gates is based on some underlying grid structure. These grids define the regions that must be routed by the detail router. MCM substrates have no such underlying structure. As a result, the bins handled by the local router may be of varying sizes and shapes.

The SURF router builds on previous approaches in several ways. Flexible bin interface specifications allow the global routing to be adjusted as more detailed local information is discovered. This information is used to increasingly refine the crossing points. By performing layer assignment and refinement during the partitioning process and not restricting the results to be one-layer one-direction, routing with fewer vias is possible.

The local routing performed at the bin level is not aimed at producing precise geometrical wiring; it is aimed at generating rubber-band sketches. The local router uses a region graph built on a constrained Delaunay triangulation to generate a rubber-band sketch. The flexible nature of the rubber-band routing allows dynamic adjustments to the routing. This flexibility can be used to alleviate local areas of congestion.

This approach represents a good balance between the need to make global decisions and satisfy constraints imposed by the more detailed local levels.

Acknowledgements

We would like to thank Kamal Chaudhary for his suggestions regarding bin partitioning and cut line selection.

This work was supported in part by the National Science Foundation under the Grant MIP-9058100, in part by Semiconductor Research Corporation under the Grant SRC-90-DJ-196, and in part by the State of California Microelectronics Innovation Computer Research Opportunities Program (proposal No. 90-037).

References

[1] R.E. Burkhard and U. Derigs. *Assignment and Matching Problems: Solution Methods with Fortran-Programs.* Springer Verlag, 1980.

[2] L.P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[3] R. Cole and A. Siegel. River routing every which way, but loose. In *Proceeding of 25th Annual Symposium on Foundations of Computer Science*, pages 65–73, 1984.

[4] Wayne Wei-Ming Dai, Raymond Kong, Jeffrey Jue, and Masao Sato. Rubber band routing and dynamic data representation. In *IEEE International Conf. on Computer Aided Design*, 1990.

[5] Wayne Wei-Ming Dai, Raymond Kong, and Masao Sato. Routability of a rubber-band sketch. In *Proceedings of the 28th Design Automation Conference*, 1991.

[6] T.C. Hu and Shing M.T. A decomposition algorithm for circuit routing. In T.C. Hu and Kuh Ernest S., editors, *VLSI Circuit Layout: Theory and Design*, pages 144–152. IEEE Press, 1985.

[7] Ulrich Lauther. Top down hierarchical routing for channelless gate arrays based on linear assignment. In *VLSI 87: VLSI Design of Digital Systems*, 1987.

[8] C.E. Leiserson and F.M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 69–78, 1985.

[9] Malgorzata Marek-Sadowska. Route planner for custom chip design. In *IEEE International Conf. Computer Aided Design*, pages 246–249, 1986.

[10] Tai-Ming Parng and Ren-Song Tsay. A new approach to sea-of-gates global routing. In *IEEE International Conf. on Computer Aided Design*, pages 52–55, 1989.