

4-1-1999

Manhattan or Non-Manhattan? - A Study of Alternative VLSI Routing Architectures

Cheng-Kok Koh

Purdue University School of Electrical and Computer Engineering

Patrick H. Madden

SUNY Binghamton, Computer Science, Science Department

Follow this and additional works at: <http://docs.lib.purdue.edu/ecetr>

Koh, Cheng-Kok and Madden, Patrick H., "Manhattan or Non-Manhattan? - A Study of Alternative VLSI Routing Architectures" (1999). *ECE Technical Reports*. Paper 39.

<http://docs.lib.purdue.edu/ecetr/39>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

MANHATTAN OR NON-
MANHATTAN? A STUDY OF
ALTERNATIVE VLSI ROUTING
ARCHITECTURES

CHENG-KOK KOH
PATRICK H. MADDEN

TR-ECE 99-6
APRIL 1999



SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Manhattan or Non-Manhattan? – A Study of Alternative VLSI Routing Architectures

Cheng-Kok Koh

School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47907-1285

Patrick H. Madden

Computer Science Department
SUNY Binghamton, Binghamton, NY 13902-6000

Abstract

Circuit interconnect has become a substantial obstacle in the design of high performance systems. A large portion of system delay, as well as power consumption and electrical noise, can be attributed to the interconnect. To address these challenges, much work has been done on topology optimization, wire sizing, driver sizing, and repeater insertion/sizing.

While these techniques have received considerable attention, the benefit can be limited in practical design applications. In modern circuits, most nets have only a few pins, allowing very little improvement through topology optimization. Wire sizing, driver sizing, and repeater insertion increase power consumption and total circuit area, and must therefore be used sparingly.

In this paper we explore a new routing paradigm which strikes at the root of the interconnect problem by reducing wire lengths directly. We present a non-Manhattan Steiner tree heuristic, obtaining wire length reductions of much as 17% on average, when compared to rectilinear topologies.

Additionally, we present a graph-based interconnect optimization algorithm, called the GRATS-tree algorithm, which allows performance optimization beyond what can be obtained through wire length reduction alone. The GRATS-tree algorithm produces for a timing-critical net, a number of possible interconnect structures that provide trade-offs among signal delay, routing area, and routing congestion.

Previously, non-Manhattan structures have been restricted to channel routing or small hand-designed applications; we present a new global router which allows large scale non-Manhattan design. Although we consider circuit place-

ments performed under rectilinear objectives, our global router can reduce maximum congestion levels by as much as 20%.

To our knowledge, this is the first work to consider extensive use of non-Manhattan structures under modern design constraints. While many challenges remain, our experiments indicate considerable promise for this approach.

1 Introduction

Interconnect optimization has become a central concern for high performance design. As circuit interconnect, contributes substantially to total circuit delay, power consumption, and electrical noise, many techniques have been proposed to overcome these problems [2]. Among these techniques are topology optimization, wire sizing, driver sizing, and repeater insertion/sizing.

In general, longer wires require such an extensive driver sizing, wire sizing, and repeater insertion that unacceptable increases in both routing area and power consumption may result. In practical design problems, however, there are a large number of signal nets for which little can be done to reduce the wire lengths. Nets with two pins cannot obtain wire length reductions through Steiner topology optimization.

In this paper, we explore the use of non-Manhattan routing architectures to reduce wire lengths of almost all nets. Non-Manhattan routing is not new—a number of channel routing approaches have shown benefit through the use of diagonal wire segments [9, 11, 12, 6], and a hierarchical Steiner tree construction for the non-Manhattan routing model has been proposed [10]. In this paper, we adopt the terminology of [10], and will use A-geometry, where A indicates the number of primary routing directions (spaced such that angles are evenly divided). 2-geometry refers to traditional Manhattan routing; 4-geometry refers to Manhattan routing with the addition of diagonal routing at 45 degrees to the X and Y axis. 3-geometry refers to a routing metric in which we have three primary axis, spaced 60 degrees apart.

While there are clear benefits in the use of non-Manhattan constructions for isolated routes, how much of that translates into the actual wire length reduction after physical layout is an open issue. Another big question is: how does a non-Manhattan architecture affect routing congestion? Coupling capacitance arising from a congested routing has a negative impact on delay, cross-talk noise and power.

In this paper, we address these two issues at the global routing level. For the feasibility study of non-Manhattan routing architectures, we have constructed a new global router that supports various A-geometry routing models. Our new global router performs routing on a general graph whose vertices are non-Manhattan routing regions. We propose the large-scale use of non-Manhattan interconnect structures, constructed through a new non-Manhattan Steiner minimal tree algorithm and a graph-based performance-driven topology optimization algorithm, called the GRATS-tree algorithm. Experimental results show that substantial wire length and congestion reductions can be obtained with

both 3-geometry and 4-geometry. To our knowledge, this is the first in-depth study on the feasibility of non-Manhattan routing architectures.

The remainder of this paper is organized as follows: In Section 2, we describe our new global router, which follows the approach of [5]. In Section 3, we describe our non-Manhattan Steiner tree algorithm, which is based on [1]. A performance driven topology generation algorithm appropriate for the proposed routing methods is described in Section 4. Experimental results, showing routing tree wire lengths for industry benchmarks are presented in Section 5. We also present results showing the performance of our global router on a variety of routing meshes, and conclude this paper.

2 Graph Based Global Routing

A major obstacle in the implementation of a non-Manhattan routing approach is in the global routing. Standard cell design usually imposes rectangular restrictions on cells and rows of cells. Macro-block design also emphasizes rectangular shapes. The regions remaining for interconnecting wires are also rectangular, resulting in predominantly rectilinear routes (with the possible exception of non-Manhattan channel routing).

Recently, the influence of these rectilinear logic blocks has been reduced. Current fabrication processes have made over-the-cell routing practical, eliminating the obstacles of standard cell rows and macro blocks. Multi-chip modules (MCMs) provide routing regions in which there are no physical obstacles. Under these new conditions, we have increased freedom, and can explore new routing paradigms.

Our new global router follows the approach of [5], in that the entire routing region is divided into a set of *tiles*, located directly above the logic elements. Global routing determines the paths taken by connections between tiles. Within each tile, a detail router determines precise routes.

2.1 Routing Paradigm

We assume that routing will occur "above" the logic elements, and that there are no rectilinear obstacles in the routing region. The lack of these obstacles allows the use of global non-Manhattan routes.

The rectilinear routing grid used in [5] is clearly not appropriate for non-Manhattan metrics, however. To support global non-Manhattan routing, our new global router supports hexagonal, *triangular*, and octagonal tilings, in addition to rectilinear tiling; these tilings are shown in Figure 2.1. For the hexagonal and triangular tilings, tiles are of uniform size and shape; for the octagonal tiling, there is a mixture of octagons and squares, allowing the entire routing region to be covered.

We focus on the number (or total width) of routes passing from one tile to another; this is the congestion. A high quality global routing will distribute connections evenly across tile borders, resulting in a decrease in the maximum

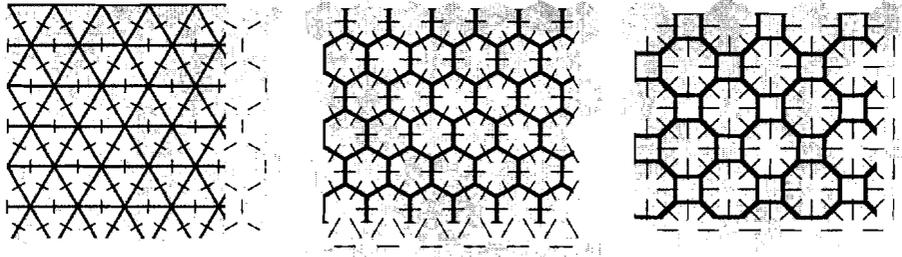


Figure 1: Three alternative tilings to a rectilinear grid. Note that the triangular and hexagonal tilings may be rotated by 90 degrees, resulting in new routing grids that prefer horizontal routing over vertical. The octagonal grid may be rotated by 45 degrees, producing "diamond" shaped tiles, rather than the square shapes shown.

congestion level. In practice, we observe that most connections are obtained with shortest path routes; average congestion levels are not increased significantly due to routing detours.

To allow comparison between routing metrics, we ensure that the lengths of tile borders are equal across the various tiling approaches. Note that this results in substantial differences in tile areas (triangular tilings have the smallest area, while octagonal tilings have a mixture of large tiles, and tiles with area equal to those of a rectilinear tiling).

Routing costs and capacities are determined on a layer-by-layer basis. The graph formulation allows this router to consider an arbitrary routing surface, with an arbitrary number of routing layers and directions. Hybrid grids, containing a mixture of rectilinear and non-rectilinear regions, or areas with differing numbers of layers and preferred directions, can be handled by the router.

2.2 Global Routing Approach

As with the global router in [5], routings are obtained by first constructing Steiner trees for each net, and then embedding the edges of each tree in the routing grid. Individual edge routings are determined by iterated rip-up and reroute, minimizing routing congestion. Steiner tree construction is performed by our modification of the algorithm of [1], and is described in the next section.

For signal nets which require higher performance than can be obtained by simply reducing wire lengths, we employ a performance driven interconnect algorithm, called the GRATS-tree algorithm. In general, there are usually several topologies that satisfy timing objectives of a given net. The GRATS-tree algorithm, described in Section 4, produces a number of possible interconnect structures for timing critical nets; these structures provide trade-offs among signal delay, routing area, and routing congestion on a general routing graph.

In practical design problems, we can expect some regions to be more congested than others. The variety of interconnect structures available for timing critical nets allows the global router a degree of freedom in the routing of these nets. As in [5], we use iterative deletion to select specific interconnect structures which obtain performance goals and also minimize congestion.

This approach overcomes the inherent limitations of traditional interconnect optimization on a single-net basis. Because only a single optimal topology is constructed for each net, any generic performance-driven global router explores a highly restricted solution space, and the congestion of any global routing solution constructed in this fashion is expected to be unacceptably high. By integrating the GRATS-tree algorithm with the iterative deletion approach of our global router, we can improve overall congestion while meeting performance goals.

3 Non Manhattan Steiner Tree Construction

Steiner tree construction for the rectilinear distance metric has been well studied! and a number of heuristics have been proposed. The 1-Steiner [7] algorithm incrementally adds a single Steiner point to an initial spanning tree, obtaining nearly 11% improvements over minimum spanning tree lengths. The edge-removal technique of Borah, Owen, and Irwin [1], which we will refer to as the "BOI" algorithm, modifies an initial MST by adding a new edge between a vertex and an existing edge, and then removing a redundant edge; this heuristic obtains tree length reductions comparable to the 1-Steiner algorithm, and has low complexity. For non-Manhattan Steiner trees, however, relatively few approaches are known; the approach of [10] initially constructs a minimum spanning tree, and then performs a hierarchical improvement process. In [8], an approach utilizing Delaunay triangulation obtained 11.8% improvements over rectilinear Steiner trees for 45° routing problems.

Our global router employs a non-Manhattan Steiner tree algorithm derived from the edge-removal algorithm of [1]. This algorithm repeatedly joins a vertex to an existing edge (introducing a Steiner point), and then removes an edge on a generated cycle. The "merge" function which joins the vertex to the edge has been modified for the new routing metrics.

While there is no equivalent to the Hanan grid for 3-geometry or 4-geometry[10], finding an optimal Steiner point to join three existing points can be done easily. Candidate Steiner points can be found at the intersections of two axis aligned vectors through two of the initial points; one of these points will be optimal. Figure 3 shows a Steiner point joining three initial points in 4-geometry; the Steiner point is located at the intersection of two vectors passing through two of the initial points. For any X based routing metric, we generate the set of candidate Steiner points (no more than $O(X^2)$ possible locations), and then select the best observed. For fixed λ , this operation is constant time. Our modification of the algorithm of [1] runs in $O(n^2)$ time per improvement pass, and require a relatively small number of passes.

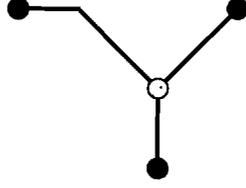


Figure 2: Steiner point location for the $\lambda = 4$ metric. The optimal Steiner point is located at the intersection of axis-aligned vectors through two of the three initial points.

4 Graph-Based RATS-Tree Construction

It is typical that nets on critical paths have a specified target delay, also known as required arrival time (RAT), for each receiver. Let q_i be the required arrival time of receiver/sink s_i . A *required-arrival-time Steiner tree* (RATS-tree) is a Steiner tree that meets the timing constraints, i.e., $q_i \geq t_i$ for each s_i , where t_i is the signal delay of s_i [4].

While RATS-tree routing on a Hanan-grid has been considered in [4], we consider routing on a general-graph in this paper. A general-graph RATS-tree formulation, which we refer to as the GRATS-tree routing, enables us to handle multi-layer routing, any arbitrary routing architecture, and congestion control more effectively. In this problem, the general-graph is the tile-based multi-layer routing graph defined by the global router introduced in Section 2. The objective is to construct a set of GRATS-trees on the given general-graph with trade-offs among congestion, routing area, and signal delays.

4.1 A Review of k -IDEA

The GRATS-tree algorithm extends the k -IDEA heuristic, a graph-based Steiner arborescence (or shortest-path tree) heuristic proposed in [3]. Given a graph $G' = (V', E')$ with source $s_0 \in V'$, we denote the shortest-path distance of $v \in V'$ from s_0 by $\Delta(v)$. The shortest-path directed acyclic subgraph (SPDAG), denoted by $G = (V, E)$, is the subgraph with $V = V'$, and directed edge $(v, v') \in E$ if and only if $(v, v') \in E'$ and $\Delta(v') = \Delta(v) + w(v, v')$, where $w(v, v')$ is the edge weight. We say that v precedes v' , denoted $v \preceq v'$, if and only if there is a directed path from v to v' in G . Moreover, $v \prec v'$ if $v \neq v'$ and $v \preceq v'$.

The k -IDEA algorithm operates on G because any arborescence of G' is a subgraph of G . The heuristic essentially follows the branch-and-bound (B&B) paradigm, albeit a restricted one. In the k -IDEA algorithm, the nodes are indexed in the increasing order of $\Delta(v_i)$. They are then processed in the reverse order of their indexes, starting from $v_{|V|}$. The algorithm maintains a *peer topology set* \mathcal{T} ; which contains a forest of subtrees constructed after node v_i is visited. The peer set \mathcal{P}_i is the root nodes of subtrees in \mathcal{T} . Let $\mathcal{X}_i = \{v' | v_i \prec v' \text{ and } v' \in \mathcal{P}_{i+1}\}$. Then, there are three possible scenarios:

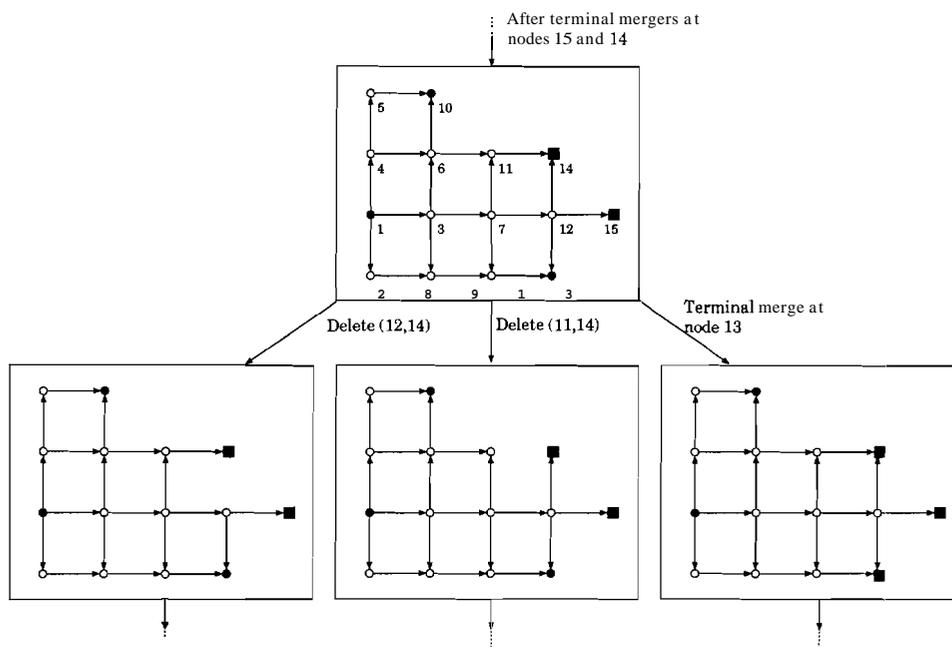


Figure 4: A snapshot of the B&B search tree illustrating the RATS-algorithm applied to the 5-terminal net in Fig. 3. Filled squares indicate processed nodes.

four nodes are 14, 16, 17, and 17 units, respectively. Essentially, edge deletion has the effect of off-loading capacitance from a critical path.

We consider deleting an incoming edge of node v after processing v . Essentially, we expand the B&B search tree by enumerating the choices of deleting an incoming edge of v . Analogous to the k -IDEA algorithm, we do not enumerate all sequences of choices of edge deletion. We restrict the number of branches due to edge deletion along a B&B search path to be no more than a user specified number, say k' .

Similar to the k -IDEA algorithm, the GRATS-tree algorithm performs a restricted B&B search and identifies the best set of skipped nodes and deleted edges. These nodes and edges are then removed from G , and the algorithm is repeated until there is no further improvement. Fig. 4 shows a partial B&B search tree corresponding to the application of the GRATS-tree algorithm to the net given in Fig. 3(a). Although the diagram appears to suggest Manhattan routing, the GRATS-tree algorithm can handle any arbitrary routing graph.

Construction of Multiple GRATS-Trees. The GRATS-tree algorithm constructs a number of GRATS-trees that trade-off among congestion, wire capacitance, and signal delay. Each (sub-)topology T generated by the GRATS-tree algorithm is associated with a triple $(Gong, Gap, Slack)$, where $Gong$ is the congestion contributed by T , Gap is the total capacitance of T , and $Slack$ is

$\min_{s, \epsilon \in T} (g_i - t_i)$.

We use three strategies to construct several high-quality GRATS-trees for each net. First, we apply the GRATS-tree algorithm on two graphs with different edge weight definitions separately. We use the distance between two connected nodes as the edge weight in one graph, and the congestion along the boundary of adjacent routing regions traversed by the edge in the other.

Second, we apply the GRATS-tree algorithm on each graph with different objective functions separately. Possible objective functions include minimizing congestion, minimizing total capacitance, and maximizing slack. For the minimization of congestion, for example, an iteration of the restricted B&B search is deemed successful only when a RATS-tree with a smaller overall congestion has been constructed.

Third, these separate applications of GRATS-tree algorithm on different graphs with different objective functions are not independent. The dependency is achieved through the maintenance of a set of irredundant topologies. We say that two GRATS-(sub-)trees T and T^1 share the same alias if they are rooted at the same node and cover the same set of sinks. T^1 is redundant if $Cong(T) \leq Cong(T^1), Cap(T) \leq Cap(T^1), Slack(T) \geq Slack(T^1)$, and at least one of the three inequalities is a strict inequality.

During an iteration of restricted B&B search process, several complete topologies are constructed. Clearly, they all share the same alias. Among these topologies, only the irredundant ones are stored. If one of these irredundant topologies improves the objective function, then the iteration is deemed successful, and the corresponding skipped Steiner nodes and deleted edges are removed for the next iteration of restricted B&B search. Moreover, we kept all irredundant sub-topologies constructed during the entire process. Whenever a merging operation results in a redundant sub-topology with respect to its aliases, we terminate the search along the corresponding path in the B&B search tree.

Clearly, we have to store a sizable number of topologies during the entire search process. To achieve a memory efficient implementation, we used the reduced tree representation introduced in [4] to allow topologies to share common sub-topologies. Other features that we have adopted from [4] include bottom-up wire-sizing optimization and bottom-up higher-order moment computation for a more accurate delay computation.

5 Experimental Results and Conclusion

To evaluate the impact of non-Manhattan routing metrics on VLSI designs, we tested our approach on a number of standard cell and MCM benchmarks. We first show the wire length reductions obtained by constructing Steiner and spanning trees for all signal nets. In Table 1, the average reduction in tree length (compared to rectilinear minimum spanning trees) is reported. While lengths under 3-geometry may exceed those of a rectilinear equivalent, we find that the average improvement across all nets is substantial. Not surprisingly, the best performance occurs under 4-geometry.

	$\lambda = 2$ BOI	$\lambda = 3$ MST	$\lambda = 3$ BOI	$\lambda = 4$ MST	$\lambda = 4$ BOI
mcc1	4.16	13.57	14.73	17.33	18.29
mcc2	0.90	12.80	13.00	16.72	17.11
mcm-test1	0.00	13.11	13.11	17.06	17.06
mcm-test2	0.00	13.36	13.36	17.51	17.51
mcm-test3	0.00	13.93	13.93	17.80	17.80
fract	2.85	1.24	3.86	7.88	9.11
struct	4.17	4.60	5.77	10.04	11.20
primary1	3.02	3.12	4.80	9.41	10.61
primary2	3.40	2.44	4.30	8.81	10.03
industry1	3.10	8.11	9.26	13.10	14.12
biomed	5.83	6.90	8.97	11.89	13.93

Table 1: Performance of non-Manhattan routing metrics; reported are the percentage reduction in wire length, relative to rectilinear ($\lambda = 2$) minimum spanning tree lengths.

It is interesting to note that these improvements are somewhat conservative: the placements have been performed under an assumption of rectilinear routing. By placing the logic elements with an appropriate cost function, even greater improvement may be obtained.

In our final set of experiments, detailed in Table 2, we report the results of our global router using a variety of tilings. For any tiling, the area of each tile varies, but the length of the border between any pair of tiles is constant; for this reason, we compare the maximum congestion on any border, and report the percentage improvement (or degradation) relative to the congestion levels observed when routing with a rectilinear grid. In experiments on rectilinear grids, our global router obtained congestion levels comparable to the global router of [5].

For 4-geometry, substantial reductions in routing congestion were obtained. This can be attributed to a combination of reduced total wire length, and also slightly larger tiles on average. For the 3-geometry, wire lengths were reduced as well; for the hexagonal grid, maximum congestion decreases compared to the rectilinear metric.

For the triangular routing grid, congestion increases for some benchmarks, and decreases for others. The smaller tile size of this grid increases the chance than a heavily congested area will be observed. If we consider the entire routing surface as a topographic map, it is clear that a sparse sampling is likely to find a lower maximum congestion value than a dense sampling.

In this paper, we do not consider routing congestion within a tile. A meaningful metric for this is difficult to construct, and in general, there is no simple method to predict if a detail router can complete a routing problem. We are quite interested in the feasibility of detail routing for non-Manhattan metrics,

	$\lambda = 3$ (triangular)	$\lambda = 3$ (hexagonal)	$\lambda = 4$
mcc1	(12.50)	8.93	7.14
mcc2	(6.90)	4.21	13.41
mcm-test1	4.00	12.00	20.00
mcm-test2	(4.44)	2.22	2.22
mcm-test3	10.77	16.92	12.31

Table 2: Percentage reduction (or increase) in maximum tile congestion relative to rectilinear (Manhattan) routing.

and are currently working on a performance driven detail router to complement our current tools.

Fig. 5 shows some topologies generated by the GRATS-tree algorithm for the global router on a hexagonal tiling. These topologies connect a 17-pin net from the benchmark circuit fract. All of them are irredundant topologies which provide a trade-off among timing slack, routing area, and congestion.

In this paper, we have presented an improved non-Manhattan Steiner tree heuristic, an algorithm for the construction of graph-based required-arrival-time Steiner trees, and a new global router appropriate for large-scale use of non-Manhattan structures.

Increased circuit sizes and performance demands drive an interest in methods to minimize interconnect delays. Non-Manhattan structures strike at the root of this problem, by reducing tree lengths for almost all nets, even those with only two pins. While the new routing metrics introduce many new problems, it also provides new opportunities for performance optimization.

Currently, our design flow is not complete; we utilize traditional placement algorithms, and are unable to apply traditional detail routing approaches. As part of our current work, we are developing placement and detail routing algorithms appropriate for non-Manhattan routing. Another important issue that deserves an in-depth study at the global-routing level is the assignment of routing direction for each layer. In 4-geometry, for example, should particular layers be dedicated to 45/135-degree wires, or do they co-exist with H/V routes? The assignment of routing directions has to make an engineering trade-off between via count and coupling capacitance of nets across layers.

References

- [1] M. Borah, R. M. Owens, and M. J. Irwin. An edge-based heuristic for Steiner routing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1563–1568, December 1994.
- [2] J. Cong, L. He, C.-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.
- [3] J. Cong, A. B. Kahng, and K.-S. Leung. Efficient heuristics for the minimum-shortest path Steiner arborescence problem with applications to VLSI

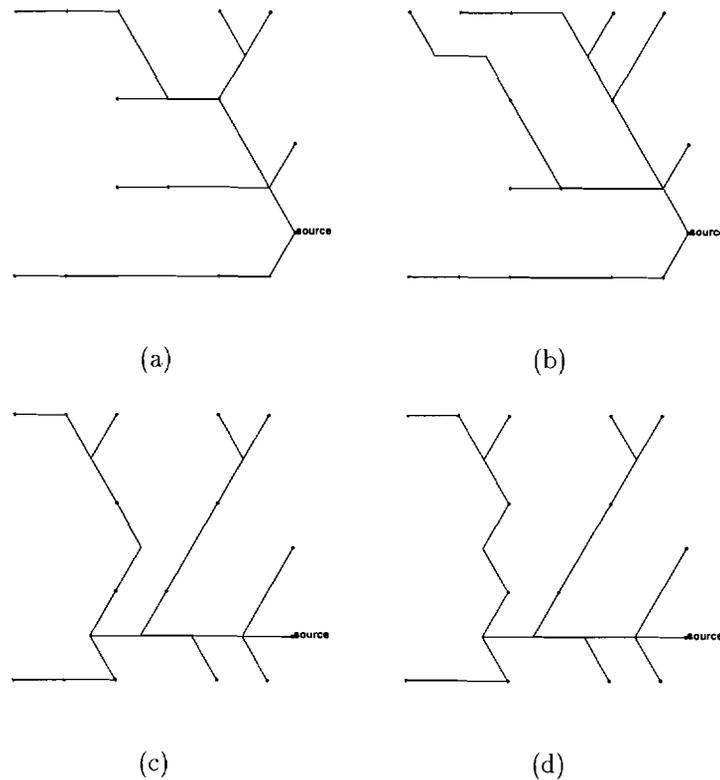


Figure 5: Topologies generated by the GRATS-tree algorithm for a 17-pin net in fract. Topologies (a) and (b) are generated from a SPDAG with node distance as edge cost, and (c) and (d) from a SPDAG with routing congestion as edge cost.

physical design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(1):24–39, January 1998.

- [4] J. Cong and C.-K. Koh. Interconnect layout optimization under higher-order RLC model. In *Proc. Int. Conf. on Computer Aided Design*, pages 713–720, 1997.
- [5] J. Cong and P. H. Madden. Performance driven multi-layer general area routing for PCB/MCM designs. In *Proc. Design Automation Conf*, pages 356–361, 1998.
- [6] S. Das and B. Bhattacharya. Channel routing in manhattan-diagonal model. *Int'l Conf. on VLSI Design*, pages 43–48, 1996.
- [7] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, July 1992.

- [8] D. T. Lee, C.-F. Shen, and C.-L. Ding. On Steiner tree problem with 45° routing. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 1680–1682a, 1995.
- [9] E. Lodi, F. Luccio, and L. Pagli. A preliminary study of a diagonal channel-routing model. *Algorithmica*, 4:585–597, 1989.
- [10] M. Sarrafzadeh and C. K. Wong. Hierarchical steiner tree constructions in uniform orientations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1095–1103, September 1992.
- [11] X. Tan and X. Song. Improvement on the diagonal routing model. *IEEE Proc.-Circuits Devices Syst.*, 141(6):535–536, December 1994.
- [12] X. Tan and X. Song. Hexagonal three-layer channel routing. *Information Processing Letters*, 55:223–228, 1995.