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

Cheng-Kok Koh
ECE, Purdue University
West Lafayette, IN 47907-1285
chengkoh@ecn.purdue.edu

Patrick H. Madden
CS, SUNY Binghamton
Binghamton NY 13902-6000
pmadden@cs.binghamton.edu

ABSTRACT

Circuit interconnect has become a substantial obstacle in the design of high performance systems. In this paper we explore a new routing paradigm that 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. Moreover, 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 two tree construction algorithms are integrated into a new global router that allows large scale non-Manhattan design. Although we consider circuit placements performed under rectilinear objectives, our global router can reduce maximum congestion levels by as much as 20%. In general we find that the non-Manhattan approach requires additional Steiner points and bends; realization of non-Manhattan routing structures requires additional vias. We observe that the increase in via cost is much less dramatic than might be expected; the benefits of wire length reduction may outweigh the additional via cost.

1. INTRODUCTION

In this paper, we explore a circuit design paradigm that departs substantially from traditional methods. We consider the extensive use of non-Manhattan structures (in contrast to current design approaches which are almost exclusively rectilinear).

Motivation for this work comes from a consideration of the current problems faced by circuit designers. Circuit interconnect contributes substantially to total circuit delay, power consumption, and electrical noise [2]; any technique which minimizes this impact is of interest.

Non-Manhattan routing is not new—a number of channel routing approaches have shown benefit through the use of diagonal wire segments [9; 12; 13; 6], a hierarchical Steiner tree construction for the non-Manhattan routing model has been proposed [11], and non-Manhattan design tools have been developed [10]. Use of non-Manhattan routing has only been on a very small scale, however; we are interested in the impact of large scale use. We have two main questions: first, what is the impact of non-Manhattan design on routing congestion, and second, what performance and wire length impact can we expect.

We have developed interconnect optimization algorithms and a new global router to allow these questions to be explored. Our design paradigm assumes that we may generate interconnection structures (at the global level) which are oriented along directions other than vertical or horizontal. In this paper, we adopt the terminology of [11], and will use λ-geometry, where λ 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. Preliminary results show substantial wire interconnect length and congestion improvements compared to traditional rectilinear routing models. To our knowledge, this is the first in-depth study on the feasibility of large-scale 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 and via estimates 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. We conclude this paper with Section 6.

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.

*This work was partially supported by Semiconductor Research Corporation under Contract #99-TJ-689.

GLSVLSI 2000 Evanston Illinois USA

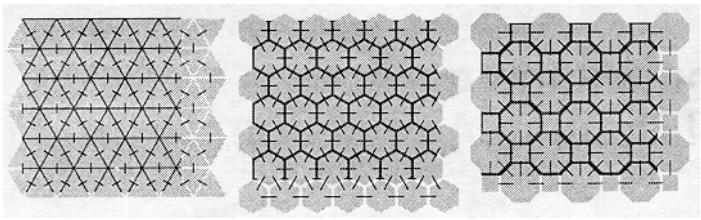

Figure 1: Three alternative tilings to a rectilinear grid. Note that the triangular and hexagonal tilings may be rotated by 90 degrees, resulting in new routing grids that prefer horizontal routing over vertical. The octagonal grid may be rotated by 45 degrees, producing "diamond" shaped tiles, rather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

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 detailed 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 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 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 delays, routing areas, and routing congestion levels 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 [11] initially constructs a minimum spanning tree, and then performs a hierarchical improvement process. For rectilinear routing, this algorithm obtains improvements of 10.3%; non-Manhattan results were not reported. 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.

3.1 Steiner Point Location

While there is no equivalent to the Hanan grid for $\lambda = 3$ or $\lambda = 4$ [11], 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 (or on top of one of the three initial points); one of these points will be optimal. Figure 3.1 shows a Steiner point joining three initial points in a $\lambda = 4$ routing metric; the Steiner point is located at the intersection of two vectors passing through two of the initial points. For any λ based routing metric, we generate the set of candidate Steiner points (no more than $O(\lambda^2)$ 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

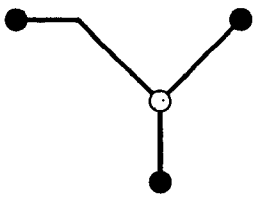

Figure 2: Steiner point location for the $\lambda = 4$ metric. The optimal Steiner point is located at the intersection of axis-aligned vectors through two of the three initial points.

$O(n^2)$ time per improvement pass, and requires a relatively small number of passes.

A sketch for a proof of this property is as follows. Assuming we have three points A, B, C, and also have optimal Steiner point S. The cost of this connection (with distance metric d_m) is $d_m(A,S)+d_m(B,S)+d_m(C,S)$. By shifting the Steiner point S along one of the directions allowed by the routing metric, this cost changes *linearly*; if all three connections require bends, we can clearly shift the Steiner point such that at least one bend is eliminated. Having eliminated a bend from one connection, the Steiner point is now located along an axis-aligned vector through one of the original points. We can now repeat the shifting process, following along this axis-aligned vector, until we encounter either one of the original points, or eliminate a bend from a second connection.

3.2 Via Insertion

Most design approaches assign a single routing direction to a layer. For example, with a Manhattan design scheme, we might have alternating layers of horizontal and vertical routing. With each change of direction, a via is required; depending on the layer assignment, the vias might span several layers. Additionally, each pin will require vias, to connect a transistor input or output on the polysilicon or diffusion layers to the layers used for routing.

These vias incur cost in terms of circuit area, resistance, and reliability. For modern design, we desire not only low wire length, but also low via counts. The use of a non-Manhattan design style clearly impacts the number of vias required; as a precise count of the number of vias required can be obtained only after detailed routing, we instead focus on the number of inserted Steiner points and bends required (both of which require via insertion). The number of layers spanned by each via is dependent on the layer assignment, and also the detailed router.

The via insertion issues are discussed in our Experimental Results section. While in general we find that the non-Manhattan approach requires additional Steiner points and bends, the difference is much less dramatic than might be expected; the benefits of wire length reduction may outweigh the cost of the additional vias.

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 general-area multi-layer routing graph defined by the global router introduced in Section 2. The objective is to construct a set of GRATS-trees, all meeting timing constraints, for a timing critical net; these GRATS-trees provide trade-offs among congestion levels, routing areas, and signal delays on the given general-graph.

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* $\mathcal{T}_i$ 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 $\mathcal{T}_i$. Let $\mathcal{X}_i = \{v' | v_i \prec v' \text{ and } v' \in \mathcal{P}_{i+1}\}$. Then, there are three possible scenarios:

- Terminal Merger Opportunity (TMO): $v_i \in \mathcal{N}$, where $\mathcal{N}$ is the set of terminals.
- Steiner Merger Opportunity (SMO): $v_i \notin \mathcal{N}$.
- No Operation: $v_i \notin \mathcal{N}$ and $|\mathcal{X}_i| \leq 1$.

If TMO applies, all the root nodes in $\mathcal{X}_i$ are merged at v_i, resulting in $\mathcal{P}_i = \mathcal{P}_{i+1} - \mathcal{X}_i + \{v_i\}$. The B&B takes place when SMO applies; either all the nodes in $\mathcal{X}_i$ are merged (as in a terminal merger) or the merging is skipped at v_i, in which case $\mathcal{T}_i$ and $\mathcal{P}_i$ remain unchanged.

The k-IDEA algorithm does not enumerate all sequences of choices between merging and skipping at each SMO. Rather, it limits the number of skipping along each branch of the B&B search tree to be no more than k. The k-IDEA algorithm performs an iteration of restricted B&B search and identifies the best set of skipped nodes that gives the minimum-weight shortest-path routing. These nodes are then removed from G, and the algorithm is repeated until there is no further improvement.

4.2 The GRATS-tree Algorithm

The GRATS-tree algorithm makes two enhancements to the k-IDEA algorithm: (i) It considers edge removal from G, on top of node deletion from G; (ii) It considers generation of several RATS-tree topologies, rather than constructing only the "optimal" topology.

Edge Deletion. Fig. 3 illustrates the need to consider edge deletion in GRATS-tree routing. The net and its corresponding SPDAG with properly indexed nodes are shown in Fig. 3(a). Fig. 3(b) shows the routing constructed by k-IDEA. Assuming a unit resistance and a unit capacitance of each wire, which is modeled as a L-type circuit, node 10 has an Elmore delay of 11 units, and nodes 13, 14, and 15 have a uniform delay of 18 units. On the other hand, if we consider deletion of edge $(12, 13)$, our GRATS-tree will construct, in addition to the topology in Fig. 3(b), a topology as shown in

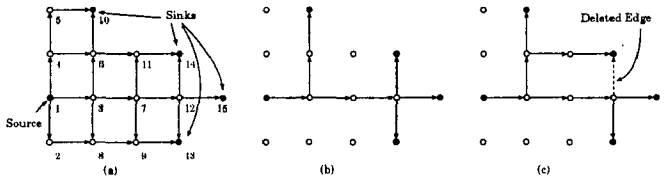

Figure 3: (a) A net and its corresponding SPDAG with properly indexed nodes. (b) The shortest-path Steiner tree constructed by k-IDEA. (c) A candidate routing topology constructed by the GRATS algorithm when the depicted edge is deleted.

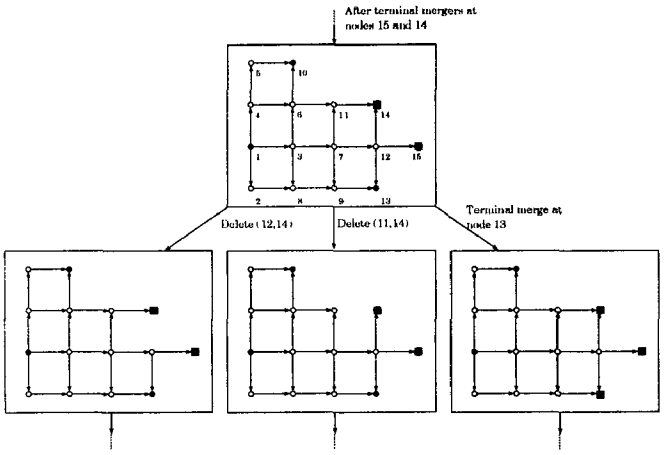

Figure 4: A snapshot of the B&B search tree illustrating the RATS-algorithm applied to the 5-terminal net in Fig. 3. Filled squares indicate processed nodes.

Fig. 3(c). The delays for the 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 levels, wire capacitances, and signal delays. Each (sub-)topology T generated by the GRATS-tree algorithm is associated with a triple $(Cong, Cap, Slack)$, where $Cong$ refers to the congestion contributed by T, Cap refers to the total capacitance of T, and $Slack$ is $\min_{s_i \in T}(q_i - t_i)$.

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 to be 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' share the same *alias* if they are rooted at the same node and cover the same set of sinks. T' is redundant if $Cong(T) \leq Cong(T')$, $Cap(T) \leq Cap(T')$, $Slack(T) \geq Slack(T')$, *and* at least one of the three inequalities is a strict inequality.

After an iteration of restricted B&B search, 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

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 (we ignore large clock nets and reset nets). In Table 1, the average reduction in tree length (compared to rectilinear minimum spanning trees) is reported. While lengths under $\lambda = 3$ 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 $\lambda = 4$.

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 second set of experiments, detailed in Table 2, we report the results of our global router using a variety of tilings. We focus on MCM routing benchmarks; in [5], routing was performed in a similar manner, and for rectilinear tilings, our new router has comparable performance. 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, which is shown in parenthesis) relative to the congestion levels observed when routing with a rectilinear grid.

	$\lambda = 2$ 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
biomed	5.83	6.90	8.97	11.89	13.93
industry1	3.10	8.11	9.26	13.10	14.12
industry2	4.17	5.29	6.24	9.80	11.16
industry3	2.54	3.39	4.08	8.60	9.50
avqsmall	1.88	5.63	5.94	9.54	10.19
avqlarge	2.01	5.21	5.63	9.28	9.93

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.

	$\lambda = 3$ (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, which is shown in parenthesis) in maximum tile congestion relative to rectilinear (Manhattan) routing. For most routing metrics, the maximum congestion level of these MCM benchmarks can be reduced.

For the $\lambda = 4$ routing metrics, 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 $\lambda = 3$ metric, 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 detailed router can complete a routing problem. We are quite interested in the feasibility of detailed routing for non-Manhattan metrics, and are currently working on a performance driven detailed router to complement our current tools.

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

Our final set of experiments, shown in Table 3 considers the number of Steiner points and bends required to implement the non-

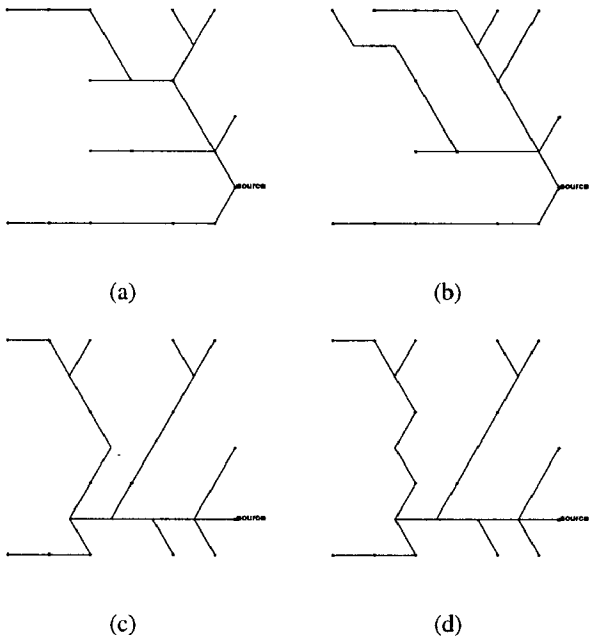

Figure 5: Topologies generated by the GRATS-tree algorithm for a 17-pin net in fract. Topologies (a) and (b) are generated from a SPDAG with node distance as edge cost, and (c) and (d) from a SPDAG with routing congestion as edge cost.

Manhattan interconnections. This provides a rough estimate of the number of vias required: in general, a via will be required at each pin, Steiner point, and change of routing direction. Note that we do not consider layer assignments, or the number of layers spanned by each via: this information is only accurate after detailed routing. Each via count entry in the table is obtained by simply summing the number of inserted Steiner points (SP), bends, and pins. It is a lower bound of the actual via count in a loose sense. While the non-Manhattan structures require (in general) more vias than the Manhattan scheme, they are comparable to the requirements of Manhattan design; routings solutions in 3-geometry and 4-geometry require 5.1% and 6.6%, respectively, more vias than in 2-geometry. Moreover, the bulk of required vias can be attributed to the inputs and outputs of transistors; these vias will be required under both routing paradigms. Furthermore, the benefits of wire length reduction may outweigh the additional via cost. In particular, the reduced wire lengths, coupled with global routing solutions that are already less congested, may reduce the number of jogs required for obstacle avoidance. As a result, fewer vias may be required at the detailed routing step.

6. CONCLUSIONS

In this paper, we have presented an improved non-Manhattan Steiner tree heuristic, a graph-based required-arrival-time Steiner tree construction algorithm, 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

		$\lambda = 2$			$\lambda = 3$			$\lambda = 4$		
Benchmark	Pins	SP	Bend	Via†	SP	Bend	Via†	SP	Bend	Via†
mcc1	2115	235	972	3322	87	1184	3386	194	1044	3353
mcc2	14659	268	6928	21855	141	7175	21975	244	7002	21905
mcm-test1	1000	0	442	1442	0	469	1469	0	440	1440
mcm-test2	1916	0	873	2789	0	911	2827	0	869	2785
mcm-test3	2508	0	1160	3668	0	1202	3710	0	1156	3665
fract	462	44	102	608	2	152	616	49	145	656
struct	5471	414	1119	7004	64	1539	7074	270	1446	7187
primary1	2941	309	517	3767	36	890	3867	260	782	3983
primary2	11226	1284	1754	14264	171	3273	14670	1092	2914	15232
biomed	17555	1985	3812	23352	526	5645	23726	1528	5141	24224
industry1	8279	1391	3788	13458	419	4895	13593	952	4377	13608
industry2	44667	5342	8924	58933	1149	14651	60467	4162	12947	61776
industry3	68093	5427	11669	85189	564	17662	86319	4366	16314	88773
avqsmall	57001	1840	7956	66797	194	10391	67586	1496	9515	68012
avqlarge	63521	2001	8582	74104	246	11260	75027	1571	10261	75344
Total Via Count†				367724			386312			391942

† The via count is obtained by summing the number of Steiner points, bends, and pins in the routing solutions. In a loose sense, it provides a lower bound of the actual via count.

Table 3: Number of pins, inserted Steiner points (SP), and bends in edges, for a variety of routing metrics. While non-Manhattan designs require larger numbers of Steiner points and bends, they are comparable to the requirements of Manhattan design. When we consider the vias required to connect the pins to the interconnect nets, the estimated increase in vias is slight.

placement algorithms, and are unable to apply traditional detailed routing approaches. As part of our current work, we are developing placement and detailed routing algorithms appropriate for non-Manhattan routing. Another important issue that deserves an in-depth study at the global-routing level is the determination of routing direction of each layer. In 4-geometry, for example, should particular layers be dedicated to 45/135-degree wires, or should they co-exist with H/V routes?

Acknowledgments

We would like to thank Dr. Naveed Sherwani, Dr. Mosur K. Mohan, and Dr. Prakash Arunachalam at Intel for their helpful discussions.

7. 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, Dec. 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 physical design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(1):24–39, Jan. 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 chanel-routing model. *Algorithmica*, 4:585–597, 1989.

[10] D. Marple, M. Smulders, and H. Hegen. Tailor: A layout system based on trapezoidal corner stitching. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 9(1):66–90, 1990.

[11] 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.

[12] X. Tan and X. Song. Improvement on the diagonal routing model. *IEEE Proc.-Circuits Devices Syst.*, 141(6):535–536, December 1994.

[13] X. Tan and X. Song. Hexagonal three-layer channel routing. *Information Processing Letters*, 55:223–228, 1995.