NonHanan Routing

Huibo Hou, Jiang Hu and Sachin S. Sapatnekar *

September 15, 1998

Abstract

This work presents a Steiner tree construction procedure, MVERT, to meet specified sink arrival time constraints. It is shown that the optimal tree requires the use of nonHanan points. The procedure works in two phases: a minimum-delay Steiner tree is first constructed using a minor variant of the SERT procedure, after which the tree is iteratively modified, using an efficient search method, to reduce its length. The search method exploits the piecewise concavity of the delay function to arrive at a solution efficiently. Experimental results show that this procedure works particularly well for technologies where the interconnect resistance dominates, and significant cost savings are shown to be generated.

1 Introduction and Motivation

In recent years, interconnect delay has become an increasingly critical factor in VLSI systems. The increased effect of interconnect resistance has played a significant role as feature sizes have entered the deep submicron range and will become more dramatic in the future. To deal with this trend, many methods have been proposed to reduce interconnect delay, among which performance-driven routing has been an active area of research. Typically, the objective of a performance-driven routing algorithm has been to minimize the source-to-sink delay using various models of abstraction. Several

*H. Hou was with the Department of Electrical and Computer Engineering at Iowa State University, Ames, IA, USA. She is now with Qualcomm Inc., San Diego, CA 92121. J. Hu and S. S. Sapatnekar are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA. This work is supported in part by the Semiconductor Research Corporation under contract 98-DJ-609, the National Science Foundation under award CCR-9800992, and by a University of Minnesota Grant-in-Aid.

types of problems have been tackled by researchers in the past, including minimizing the average source-to-sink delay over all sinks, minimizing the maximum source-to-sink delay, and minimizing a weighted sum of the delay from the source to some critical sink(s).

In the situation where the interconnect is purely capacitive and has negligible resistance, the minimum delay tree is identical to the minimum length Steiner tree, since minimizing the length is equivalent to minimizing the delay.

When the resistance of the interconnect is appreciable, the type of route is determined by the stringency of the timing specifications. An appropriate problem formulation for the timing-driven routing problem is as follows:

$$\begin{array}{ll} \text{minimize} & \text{Total tree length} \\ \text{subject to} & \text{Delay(sink } i) \leq T_{spec}(i) \text{ for } i \in \text{List of sinks} \end{array}$$

If the timing specifications are very loose, then the solution to this problem would be to minimize the total tree length, and would result in a minimum-length Steiner tree. If the timing specifications are extremely tight, and the resistance of the driver is much less than the resistance of the interconnect, a star-like topology is generated. The star-like topology emanates at the root of the tree and has direct connections from the root to each sink; this corresponds to the maximum length for the tree. For intermediate specifications, an intermediate solution between the minimum length Steiner tree and the star-like topology is optimal. This is illustrated in Figure 1.

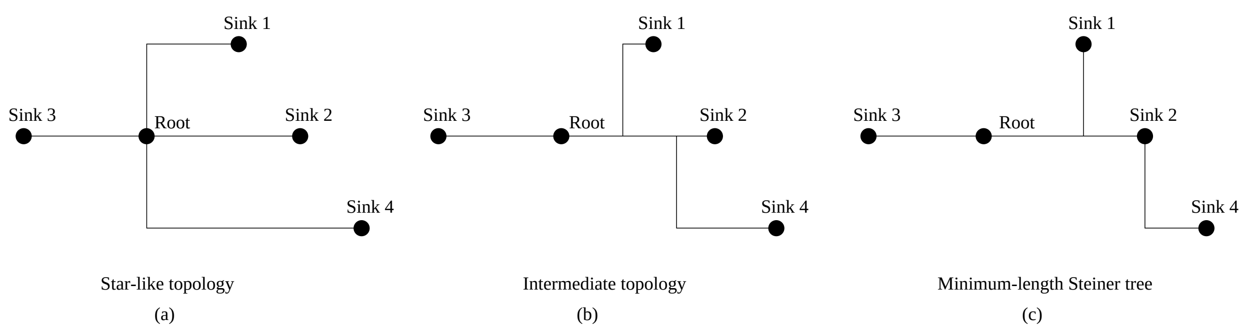

Figure 1: Three scenarios for building a Steiner tree.

The task of delay optimization of a circuit consists of appropriately optimizing the gates and optimizing the interconnect wires. The gates may be optimized through actions such as mapping the circuit to a set of complex gates (possibly from a library) and/or sizing the gates to achieve the requisite driving power. For optimal circuit design, the cost of the optimization must be borne equally by the gates and by the interconnects in an interconnect-dominated environment.

Therefore, if the timing-driven routing solution results in a star-like interconnect, it is likely that more could be done in optimizing the gates to reduce the stringency of the specifications on the interconnect. Similarly, if the result is a minimum-length Steiner tree, it is likely that if the specifications on the gates were made less stringent, a greater share of the burden could be borne by the interconnect, and the cost of gate-level optimizations could be reduced. This implies that in many cases, the result of the routing is more likely to be the intermediate case in Figure 1(b) than either of the extremes in (a) or (c).

In this paper, we develop a performance-driven routing formulation whose objective is to meet a specified delay constraint at each sink. The approach can also be generalized to solve the problem of minimizing the maximum delay over all sink nodes. It is shown here that in contrast to conventional minimum-length Steiner trees, this problem can have an optimal solution that requires the introduction of Steiner points that do not lie on the Hanan grid [7]. This happens in situations described by the scenario in Figure 1, which, as we have just argued, are likely to occur in well-designed circuits where the task of delay reduction is equally borne by the gates and the interconnects.

Early work solved the problem of building performance-driven routing topologies by solving the minimum-length Steiner tree routing problem, a valid approach in the times when the resistive effects of interconnect were negligible. After Hwang's work [11], which proved that the ratio of the cost of a rectilinear minimum spanning tree (MST) to that of a minimum length rectilinear Steiner tree (RST) is within a factor of 3/2, many heuristic algorithms [8–10, 12, 16] used the MST as a starting point to derive the RST. For example, the algorithm in [10] minimizes the cost of RST by maximizing the overlaps between the layouts in a existing separable MST. Kahng *et al.* [13] developed the iterated 1-Steiner heuristic method, which introduces only one Steiner point into a net at one time such that the current minimum spanning tree cost is minimized. A good overview of these techniques is provided in [6, 14].

However, in the deep submicron range, the interconnect capacitance and resistance have become comparable to, or dominant over, the gate capacitance and the output driver resistance, and cannot be ignored during delay calculation any more. Therefore, it is apparent that for leading-edge technologies, minimum net lengths do not always yield minimum delay. Thus, performance-driven methods are introduced to minimize the source-to-sink delay. The A-tree algorithm [5] and SERT [3] use the Elmore delay model, with a justification of its fidelity in [1] being provided as a basis for doing so. SERT is based on a greedy algorithm that optimizes the Elmore delay directly as the routing tree is constructed. Significant improvements over existing performance-driven routing tree constructions were demonstrated in this work. However, the area overhead of SERT is discouragingly large and for some technologies, it tends to generate star-like topologies due to its greedy nature. In [17], a new method was proposed, which was a departure from the constructive greedy heuristics. The procedure builds routing topologies induced by sink permutations and then maps the topologies to a routing layout. The algorithm derives an area/delay tradeoff under the Elmore delay model and incorporates simultaneous wire-sizing by using dynamic programming. Other approaches to timing-driven Steiner tree construction include [4, 15].

All of the above techniques have a common bottom-line: they are based on Hanan's theory [7], which showed that if one were to draw horizontal and vertical grid lines through the source and sinks of the given signal net, one could be assured of the existence of a minimum-length rectilinear Steiner tree, all of whose Steiner points lie on the intersection points in the resulting grid. Thus, the possible Steiner points can be chosen from a finite set, namely, the so-called Hanan grid. In [3], it was proved that only points on the Hanan grid need be considered while solving the problem of minimizing the weighted sum of critical sink delays. For the minmax problem of minimizing the maximum sink delay, it was shown in [2] that it is possible to build a better solution by considering points off the Hanan grid, but it was stated that such situations are uncommon and can be ignored. In this work we show that it is possible in cases to arrive at significantly better solutions by considering nonHanan points during Steiner tree construction for two problems:

(a) the minmax problem, and

(b) the problem of achieving a specified delay at each sink node.

It should be pointed out that the problem (b) above can be transformed into the form of problem (a).

Example 1: We illustrate a simple example showing that a nonHanan node is required to minimize the maximum source-sink delay during tree construction. Consider a net with a source at (0,0) and two sinks, a and b, at (3,0) and (1,4), respectively. We assume, for simplicity, a unit resistance and a unit capacitance per unit length. The driver has a source resistance of 6, and the sinks a and b have load capacitances of 1 unit and 4.5 units, respectively. The delays are calculated here under the Elmore delay model, described in Section 2.1. The variation of the delay at each sink as the Steiner point x is moved from (0,0) to (1,0) is shown in Figure 2[1]. The maximum sink delay for the tree is minimized at x = 0.33[2].

This example illustrates that in real design problems, the timing requirements at different sinks are often contradictory and it is necessary to arrive at a solution where all of the sinks are considered together.

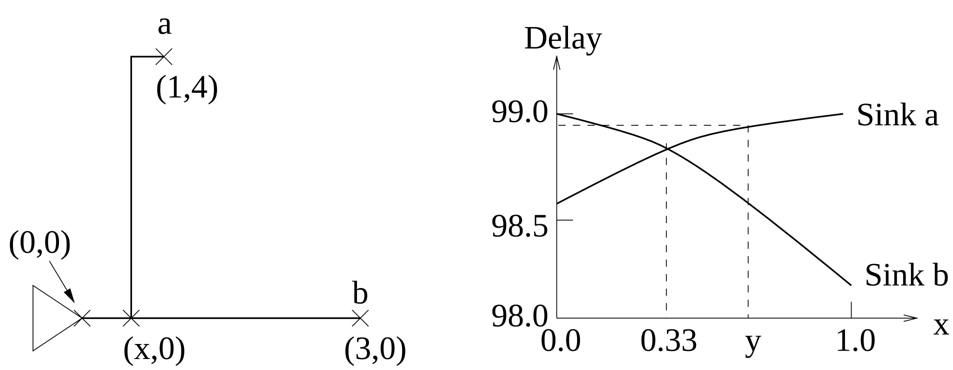

Figure 2: Illustrating the effects of nonHanan Steiner points

For the problem of achieving a specified timing constraint T_{spec}, it is also easy to show that the optimal solution may lie at a nonHanan point. Any procedure that restricts of Steiner points to Hanan points alone would lead to a larger than optimal tree cost. Therefore, for the problems of achieving a set of specified sink delays, and of minimizing the maximum source-sink delay, the best Steiner points do not necessarily lie on the Hanan grid.

However, the routing problem is clearly more difficult when the number of Steiner points becomes infinite, and the search space becomes extremely large. This paper utilizes the properties of the delay function to arrive at a simple and efficient method to overcome this challenge.

[1]The logic in [1] can be extended to show that a Steiner point to the right of (1,0) is suboptimal.

[2]We caution the reader not to be unduly swayed by the modest delay reductions in this simple example, as our experimental results will show that it is possible to achieve more significant improvements on other larger examples.

2 Preliminaries

2.1 Delay model

The delay in this work is modeled using Elmore delays [19], which are briefly described here. Given a routing tree $T(N)$ rooted at the source n_0, let e_v denote the edge from node v to its parent in $T(N)$. The resistance and capacitance of edge e_v are denoted by r_{e_v} and c_{e_v}, respectively. Let T_v denote the subtree of T rooted at v, and $C_{d,v}$ denote the downstream tree capacitance of T_v, which is the sum of sink and edge capacitances in T_v. The Elmore delay from the predecessor of node v along edge e_v is equal to $r_{ev}(c_{ev}/2 + C_{d,v})$. This procedure can be used to recursively compute the delays. Given that the root node is driven by a driver of resistance r_d, the Elmore delay $t_{ED(n_i)}$ at sink n_i is calculated as:

$$t_{ED}(n_i) = r_d C_{no} + \sum_{\text{all fanouts}} r_{ev}\left(\frac{c_{ev}}{2} + C_{d,v}\right) \tag{1}$$

2.2 The SERT method

In this section, we provide a brief overview of the SERT (Steiner Elmore Routing Tree) algorithm in [3]. Before proceeding further, we define the following terms:

Definition [3]: A *segment* of a tree T is defined as a contiguous set of straight edges in T that are either all horizontal or all vertical. A *maximal segment* is a segment that is not properly contained in any other segment. The *root* of a segment is the extremal node of the segment that is closer to the root n_0 of the tree.

The essential idea of the SERT method is based on building a greedy Steiner tree using an approach similar to Prim's algorithm. Starting with a trivial tree consisting of only the source n_0, the tree is iteratively built by adding a pin n_i in the tree and a sink n_j outside the tree so that adding edge (n_i, n_j) yields a tree with the minimum Elmore delay. The iterations continue until all sinks have been included in the tree. The algorithm is predicated on two results:

(a) Selecting n_i to be downstream of the closest connection (CC) from n_j to the partial tree will yield a larger delay than connecting it to the CC.

(b) For a maximal segment, if x is the distance from its root m_0 to the point n_i in the partial tree, then it was shown that the Elmore delay to any sink is a concave function of x when the connection point n_i lies between m_0 and CC, both inclusive. Therefore, any weighted sum of the Elmore delays to a fixed set of critical sinks is a concave function of x and is minimized by setting n_i to be either the root m_0 or CC.

These observations reduce the search space considerably as m_0 and CC are the only candidate connection points for a segment in each iteration. As both m_0 and CC lie on the Hanan grid, all Steiner points in the SERT tree must necessarily lie on the Hanan grid. If the path from a CC to the root of the tree consists of multiple maximal segments, then this property is true on each maximal segment, and the delay as a function of the distance from the root of the tree is therefore a piecewise concave function, with each derivative discontinuity corresponding to some segment edge.

Note, however, that the concavity logic does not extend to the use of minmax formulations or formulations where the delay at each sink is a constraint. This is due to the fact that while the positive weighted sum of concave functions is concave, the maximum of concave functions is not concave, as seen from the counterexample presented in Figure 2.

3 The MVERT algorithm

3.1 Problem formulation

We now describe a procedure for building a minimum cost tree that satisfies the delay constraints at each sink. Unlike previous approaches, this technique permits the possibility of a Steiner point that lies off the Hanan grid. The procedure described in this work is referred to as the Maximum delay Violation Elmore Routing Tree (MVERT) algorithm. A basic concept that is used in this work is the idea of a delay violation, which is defined as the amount by which the delay $d(n_i)$ at sink i violates its specification, i.e.,

$$Violation(i) = d(n_i) - T_{spec}(i) \tag{2}$$

Clearly, a positive value of the violation implies that the constraints could not be met. A large negative value of the violation, on the other hand, indicates the possibility of overdesign, and it is possible in some cases to reduce the cost of the Steiner tree by bringing the violation value to be closer to zero. This idea motivates the formal statement of the MVERT problem as follows:

MVERT Problem: Given a signal net N = $\{n_0, n_1, ..., n_k\}$ with source n_0, construct a Steiner routing tree $T(N)$ such that the total length of the net is minimized while the delay violation at each sink node is less than 0, i.e.,

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{\text{all segments } k} l_k \\ \text{subject to} \quad & d(n_i) \leq T_{spec}(i) \text{ for } i = 1, 2...n. \end{aligned} \tag{3}$$

where l_k is the length of each segment in the resulting tree.

3.2 Properties of the formulation

The MVERT problem formulation is quite general in nature. If the value of T_{spec} at each node is specified to be zero, then the problem is identical to the maximum source-sink delay problem. The user may specify different delay specifications at different sinks as desired.

Theorem 1: Consider any Steiner tree T connecting a source n_0 to sinks n_1, n_2, ..., n_m. Consider a node n_j, and let it be connected to the tree at an upstream point n_k along a maximal segment s of the tree, where n_k is possibly a Steiner point. Let T' be the subtree rooted at n_j, let CC be a closest connection connecting n_j to the tree $T \backslash T'$ along maximal segment s' (with s' being possibly the same as s), and let m_0 be the root of s'. Let T_1 be the tree formed by connecting n_j to $T \backslash T'$ at CC. Then

(a) The tree obtained by connecting n_j to $T \backslash T'$ at a point downstream of CC on segment s' results in a tree with a larger length and larger delays at each sink than T_1.

(b) If x is the distance from m_0 to any point n_i in the partial tree, then the Elmore delay from the root n_0 of the tree to any sink is a concave function of x when the connection point n_i lies between m_0 and CC, both inclusive.

Proof: Parallel to that of a similar result in [3].

3.3 Tree construction procedure

Since the restriction to a Hanan grid is no longer valid, the set of candidate Steiner points is infinite, and it is necessary to find an efficient method to identify the best Steiner points. The MVERT algorithm is divided into two phases:

(a) The initial tree construction phase, where an initial tree is heuristically built to minimize delay.

(b) The cost-improvement phase, where the tree is iteratively refined to reduce its cost while ensuring that it meets all timing specifications.

3.3.1 Phase 1: Initial tree construction

The first step in tree construction is similar to the ERT construction procedure in [3], with the difference that we minimize the maximum delay *violation* rather than the maximum delay. This procedure considers only candidate points on the Hanan grid, and is described as follows:

Input: `A signal net` N `with source` n_0 `and sinks` $n_1, \cdots n_k$

Output: `A heuristic minimum-delay routing tree` T_1

`Initial tree` T_0: $(V, E) = (\{n_0\}, \emptyset)$

`while` $(|T| < N)$ `do`

`Find` $u \in V$ `and` $v \notin V$, `to minimize the maximum Elmore delay violation from` n_0

    `to any sink in the tree` $(V \cup \{v\}, E \cup \{(u, v)\})$

$V = V \cup \{v\}$, $E = E \cup \{(u, v)\}$

`Output resulting routing tree` $T_1 = (V, E)$

Another difference between the SERT algorithm and this procedure is that in each iteration, for each node outside V, we attempt connections with every possible maximal segment, rather than with only the closest maximal segment.

3.3.2 Phase 2: Cost improvement

The initial tree constructed above attempts to minimize the maximum delay violation at any leaf node. Since the criticality of a sink is dependent on the timing constraints and the positions of the other sinks, this provides a natural technique for automatically detecting critical sinks and providing more importance to them in the tree construction process.

However, the tree so constructed may be conservative as its objective is to minimize the violation, rather than to simply ensure that the constraints are never violated. Moreover, the Phase 1 procedure considers only Hanan grid points as candidate Steiner points. Therefore, it attempts to connect each point either to the closest connection (CC) in the partial tree, or to the root node m_0 of a maximal segment. If the delay violation associated with a CC connection were larger than the delay associated with a connection to m_0, then the algorithm would make a connection to m_0. However, due to the interactions between paths the MVERT solution may lie at a different (and possibly nonHanan) point, and the connection to m_0 results is a larger net length than is necessary. Therefore, we examine the tree constructed in Phase 1 and move node connections from the root of a segment towards CC in a bid to reduce the tree length, as shown in Figure 3, while ensuring that all timing constraints are satisfied.

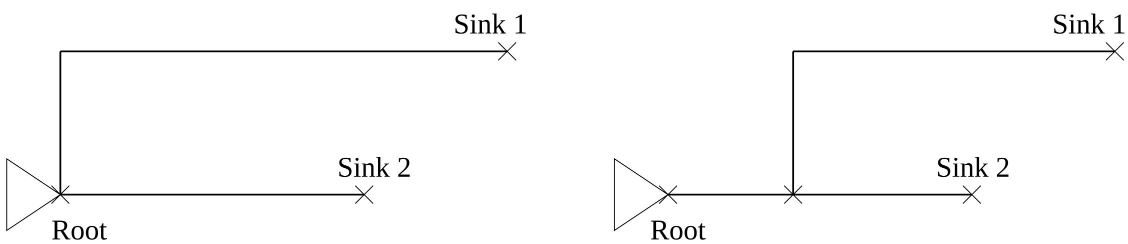

Figure 3: Creating a nonHanan Steiner point that meets timing specifications

Thus, in Phase 2, we refine the tree built in Phase 1 to reduce its length while ensuring that the timing constraints are satisfied. The pseudocode for this procedure is shown below:

Input: `Routing tree` $T_1 = (V, E)$ `from Phase 1`

Output: `An optimal routing tree` T_2

`if (`T_1 `did not satisfy constraints) OR (all sinks were connected to` CC `in` T_1`)`

        $T_2 = T_1$`; exit`

`Sort sinks` v_1`,`v_2`,` v_3`...  that are not connected to a` CC `in` T_1 `in descending`

    `order of the distance to` n_0

`for (each such sink` v_i `) do` **Improve_Cost**`(`v_i`)`

`Output resulting routing tree` $T_2 = (V, E)$

If the minimum-delay tree, T_1, did not satisfy the given timing constraints, then it is assumed that any transformed version of the tree created by altering the topology will not help meet the constraints. Secondly, if the minimum-delay tree, T_1, only has connections to CC points, then it is assumed that no further improvements are possible in this phase. In either case, the output of this phase will be the tree T_1[3]. However, if neither of the two conditions hold, then it is possible to improve the cost of the tree. The idea is illustrated in Example 1 for the constraint of 98.8 units, where we see that a connection to (y,0) is preferable to a connection to (0.33,0). In such a case, the **Improve_Cost** routine is invoked to reduce the cost of the tree T_1. The routine processes sink nodes, processing the one farthest from the root first as follows:

Function **Improve_Cost**(v_i)

Find the path p from n_0 to v_i, Q = $\{u_1, u_2, \cdots, u_k\}$, where u_i is a sink

    on the path between n_0 and v_i.

For each (u_i connected to $m_0 \neq CC$) do

    Remove connection of u_i to m_0, while keeping the connection of u_i to its

        downstream nodes.

    Define $T' = T \backslash \{(e_i) \cup (T_{u_i})\}$, where e_i is the edge that connects u_i to m_0,

        T_{u_i} is the downstream tree of u_i in T.

    Reconnect(u_i, T');

The function **Reconnect**(e_i, T') reconnects the node u_i to the tree T' at the nearest point to CC where constraints are satisfied.

3.3.3 Finding the optimal reconnection point

Consider the set of constraints on the routing tree from Equation (3). Rewriting them in the form $d(n_i) - T_{spec}(i) \leq 0$ for all sinks i, we see that the maximum violation must always be nonpositive. Since each of the $d(n_i)$'s is a concave function of the connection point x by Theorem 1, and since any concave function shifted by a constant is a concave function, this implies that we must find a

[3]Note that this is only an assumption and not a guarantee, since the Phase 1 tree is heuristically constructed.

reconnection point x such that the maximum of the set of concave functions is nonpositive. This is pictorially shown in Figure 4 below for a net with four sinks, v, w, y, and z; the maximum violation function is shown by the darkened line. This piecewise concave function is composed of three concave *pieces*. Note that the graph shows that sink z is never critical in this case, for any value of x. The delay violation at each sink as a function of x is a concave function and the objective is to find the value of x that is closest to CC (corresponding to a minimal increase in the net length) that satisfies all constraints. In Figure 4, this point is found to be x^*. This point would, in general, be a nonHanan point.

Corollary 2: The search space for the reconnection point in the pseudocode in Section 3.3.2 can be restricted to the interval between the root m_0 of each maximal segment, and the closest connection point, CC, on that maximal segment.

Proof: This follows directly from Theorem 1.

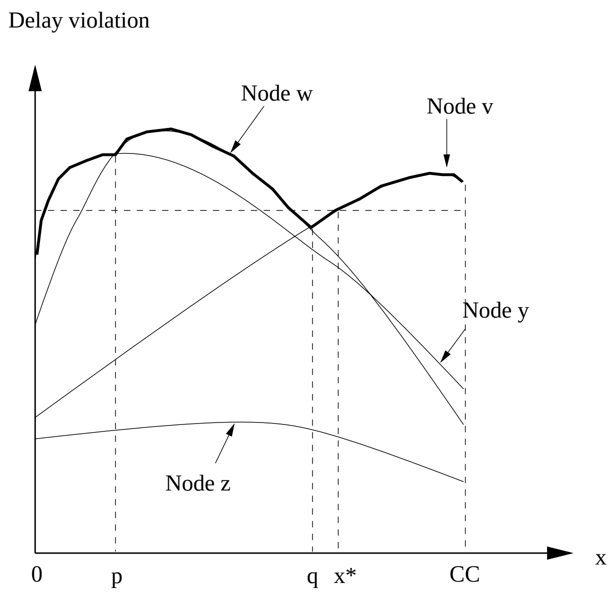

Figure 4: Finding the optimal value of x that satisfies the timing constraints

In searching for x^*, we observe that it is possible to perform a search on the value of x from 0 to CC, while taking advantage of the fact that the value on each concave piece is minimized at its intersection with the concave piece on either side (if such a piece exists), or at 0 or CC otherwise. In Figure 4, this translates to the fact that for the minmax problem, the only candidate solutions are 0, p, q and CC. This permits a reduction of the search space from the infinity of points between 0 and CC.

For the problem of meeting timing specifications at each sink, several pruning strategies are possible for the search.

- If the value of $\max[d(n_i) - T_{spec}(i)]$ at each end of the concave piece is positive, then the search finds the end point of the concave piece away from CC, and continues from there. This is justified by the fact that each point on the concave piece must necessarily have a positive violation whenever the end points of the piece have a positive violation.

- Consider a binary search on a concave piece with end points x_1 and x_2 ($x_1 < x_2$) with values $f(x_1)$ and $f(x_2)$, respectively. If $T_{spec} > f(x_1)$, $T_{spec} < f(x_2)$ and $T_{spec} < f(\frac{x_1+x_2}{2})$ as illustrated in Figure 5, then the search can completely eliminate the interval $[\frac{x_1+x_2}{2}, x_2]$. This follows from the fact that any a concave function over an interval is concave over any continuous subinterval. By a symmetric argument, if $T_{spec} \geq f(\frac{x_1+x_2}{2})$, then the search is reduced to the interval $[\frac{x_1+x_2}{2}, x_2]$.

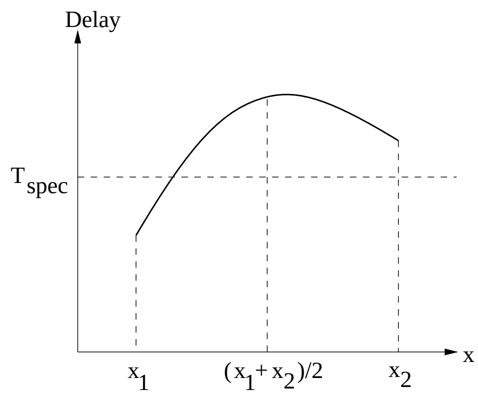

Figure 5: Using piecewise concavity to enhance the optimization procedure

The pseudocode corresponding to this search is shown below:

Function **Reconnect**(u_i,T')

Find CC for u_i in T'.

Find the path from the root, n_0, of the tree to CC. Let $n_1, n_2 \cdots, n_l$ be the
    Steiner nodes/sink nodes along this path, arranged in reverse
    topological order from n_0.

Divide the path into maximal segments p_1 = (n_0, n_1), p_2 = (n_1, n_2), p_3 = (n_2, n_3),

```
    ..., p_l = (n_{l-1}, n_l).
i = l; current_segment = p_i;
while (TRUE)
    Perform a search to find the closest point to CC on current_segment
        that satisfies the delay constraints
    If such a point exists
        choose it as the connection point; return(success);
    else
        i--;
    if (i > 0) then current_segment = p_i; else return(failure);
```

The search must be performed along each maximal segment individually since the presence of a sink node or a Steiner point branching off to another node leads to a potential discontinuity in the derivative of the delay function. The efficiency of the search can be greatly enhanced by taking advantage of the piecewise concavity of the delay function, as described earlier in this section.

The search for the best connection begins at CC. If the value of the delay violation at CC is negative, then we are done; otherwise a step of size L, a user-defined parameter, is taken to go from the current point, σ_1, to the next point, σ_2. Note that by construction, there will always be a delay violation at σ_1. After each step, the following possibilities must be considered:

(1) If the maximum delay violations at σ_1 and σ_2 correspond to the same sink, then the points lie on the same concave piece of the maximum delay violation function. In this case, two scenarios must be considered:

   (a) If both delay violations are positive, then the function must necessarily be positive over the entire interval between the points σ_1 and σ_2, since the delay violation function is a maximum of concave functions. Therefore, we set $\sigma_1 \leftarrow \sigma_2$ and continue the search.

   (b) If the delay violation at σ_2 is negative, then the optimum connection must lie between the points corresponding to σ_1 and σ_2. Since the delay violation function is continuous and its value at two extremes of the interval are of opposite signs, the function value must be zero at some point within the interval. Therefore, we continue the search between these points to identify the closest point to CC with a zero violation.

(2) If the maximum delay violations at σ_1 and σ_2 correspond to different sinks, then we perform a binary search to find the intersection point of the concave piece on which σ_1 lies, with its immediate neighbor in the direction of the root. During this binary search, we perform the tests listed in condition (1) if the two points both correspond to the concave piece of σ_1. In practice, this is not likely to be a significant problem since the number of concave pieces in the delay violation function is $O(k)$.

3.3.4 Complexity issues

We now conduct a rough complexity analysis of the procedure. For the first phase where the heuristic minimum-delay tree is built, the procedure is similar to that of the ERT procedure in [3]. The complexity of our method is $O(k^4)$, where k is the number of sinks. The reported complexity of this procedure in [3] is $O(k^3)$, and the discrepancy is due to a difference in the manner in which a sink is added to the partial tree in each iteration. Instead of finding the geometrically closest connection for each node outside the tree as in [3], our approach attempts to create closest connections to *all* possible maximal segments in the partial tree, and evaluates their delay to choose the connection to be made in each iteration. The number of maximal segments is $O(k)$ (since the addition of each new vertex introduces at most two new maximal segments in the tree), and this leads to the increased complexity.

In the second phase, each sink is processed precisely once, either in the main routine, or in function **Improve_Cost**. The cost involved in processing a sink is in finding a reconnection point, if required. The function **Reconnect** processes all maximal segments on the path to the root; the number of maximal segments, as argued earlier, is $O(k)$. For each such segment, the cost of the search for a connection point is $X \cdot k$, where X is the number of binary search iterations, and each such iteration requires a delay calculation that necessitates an $O(k)$ traversal of the circuit. Then the cost of the nonHanan routing procedure is $O(k^4 + k^2 \cdot (k) \cdot X)$.

In this analysis, we have deliberately used the letter X to denote the cost of the search since this cost is rather unpredictable. In the best case, if both ends of the segment lie on the same concave piece, one computational step is required. In the worst case, if the delay violation

Table 1: Results on technology 1 (IC)

Circuit	Cost		
	After Phase 1	After Phase 2	Percentage Improvement
Net 1_IC	8830	7270	21.4%
Net 2_IC	10490	9540	10.0%
Net 3_IC	8200	7580	8.2%
Net 4_IC	9360	8130	15.1%
Net 5_IC	6450	5630	14.6%
Net 6_IC	7450	5250	41.9%

function on the segment contains t concave pieces, the cost incurred will correspond to identifying the intersection of each of these pieces with its neighboring concave piece. Since the number of concave pieces in the maximum delay violation function is upper-bounded by k, the number of sinks, at most $O(k)$ such intersections will need to be determined during the entire search. If the binary search is continued until the size of the interval is reduced to ϵ, then the cost of each such search requires $O(\frac{L}{\epsilon})$ search points. Therefore the worst-case complexity of the procedure is $O(k^4 + k^3 \cdot (k) \cdot \frac{L}{\epsilon})$.

4 Experimental results

The MVERT algorithm was applied to nets in two technologies: an IC technology and an MCM technology. Design parameters for each technology are taken from [3]. The results are shown in Tables 1 and 2, respectively. Each of the examples shown here is a five-pin global net, and the cost is measured as the sum of wire lengths, in microns.

It is worth mentioning here that the topologies were generated randomly and the timing constraints were chosen artificially since no benchmarks are available. The procedure for choosing the timing constraints was as follows: the minimum-delay Phase I tree was built and the delays at each sink were found. The constraints to sinks connected directly to the source were then set to be larger than the actual delay after Phase I. It should be mentioned that if large changes are made here, the Phase I tree may change (since it orders the node in terms of the largest delay violation, which is dependent on the constraint value). Therefore, it is not guaranteed that a change in the constraint will lead to nonHanan points; in fact, in some cases, we found that an alternative

Table 2: Results on technology 2 (MCM)

Circuit	Cost		
	After Phase 1	After Phase 2	Percentage Improvement
Net 1_MCM	6700	6330	5.8%
Net 2_MCM	6590	6000	9.8%
Net 3_MCM	7330	6490	12.9%
Net 4_MCM	7340	4250	72.7%
Net 5_MCM	6960	5680	22.5%
Net 6_MCM	7600	7260	4.7%
Net 7_MCM	7920	6540	21.1%
Net 8_MCM	8120	6760	20.1%
Net 9_MCM	8110	7630	6.3%
Net 10_MCM	6690	6050	10.6%
Net 11_MCM	8780	5950	47.6%
Net 12_MCM	8190	7310	12.0%
Net 13_MCM	8980	6930	29.6%
Net 14_MCM	6190	5130	20.7%
Net 15_MCM	7600	6660	14.1%
Net 16_MCM	7940	6850	15.9%

topology was generated in Phase I using Hanan points only.

It is seen that in each case, for the nets shown and the timing constraints chosen, significant reductions in the cost function are permitted by this procedure. Overall, we observed that better improvements are achievable for technology 2 than for technology 1. The explanation for this stems from the fact that the driver resistance is lower in technology 2, due to which the resistive effects of interconnect are more pronounced. Under such a situation, a SERT-like algorithm (or Phase 1 of our procedure) is more likely to generate star-like connections that connect more nodes to the root node. Since many of these connections may, under certain timing constraints, be overkill, in Phase 2 these connections are moved to be closer to CC. This has the added benefit of reducing the routing congestion near the root node.

Specifically, in the case of Net 4 in technology 2, it is interesting to observe the routing topology before and after Phase 2, as shown in Figure 6. Note that the picture is not drawn to scale, but shows the essential idea that an initial star-like topology is reduced to a nonminimum-length Steiner tree.

A comparison of the delays with SPICE simulations was made for six nets that were randomly

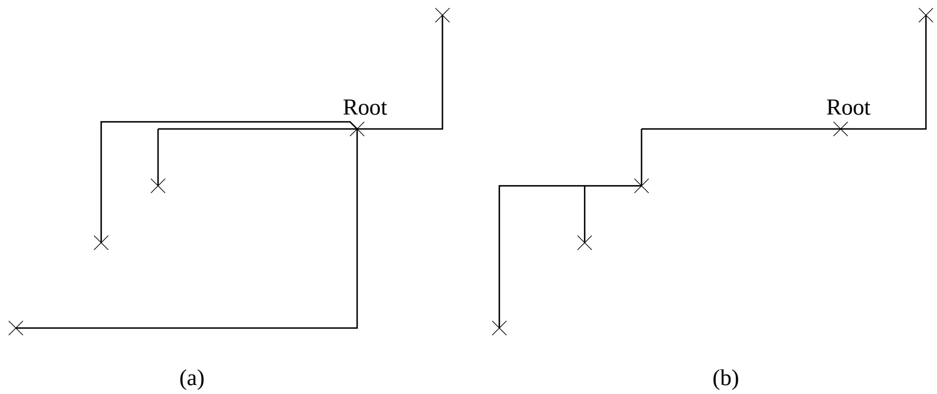

Figure 6: Net4_MCM5 after (a) Phase 1 (b) Phase 2

chosen from the 22 nets presented in Tables 1 and 2. It was seen that the discrepancy between the SPICE delay and the Elmore delay (multiplied by $log2$) was under 20% at each sink for these trees. However, it is true that the Elmore delay is not guaranteed to be close to SPICE delays, and indeed, is often not so. In our future work, we intend to address the extension of this procedure to higher-order delay models.

The CPU times for the procedure are shown in Figure 7. The y axis shows the average CPU time over several nets with the same pin count and the relationship illustrates the practical computational complexity. Since a 20-pin net can be handled in about fifteen seconds, this also shows that the procedure can realistically be applied to global nets of a reasonable size. It should be emphasized that this picture is only qualitative since the CPU time can vary depending on the specifications (it depends, for example, on the number of nets to be moved in Phase 2). However, the graph of the averages illustrates two points: firstly, the nonlinear variation of the CPU time with the number of pins, and secondly, that even a 20-pin net can be handled in a reasonable amount of time.

5 Conclusion

A new technique for finding a minimum cost Steiner tree subject to timing specifications at the sink nodes has been presented. It has been shown that the use of nonHanan Steiner nodes can provide noticeable benefits in improving the cost of the tree. If the timing specifications at the sinks are very loose, a minimum length Steiner tree is the correct solution, and if the timing specifications are very tight, a minimum-delay Steiner tree is desirable. It is in between these two extremes that

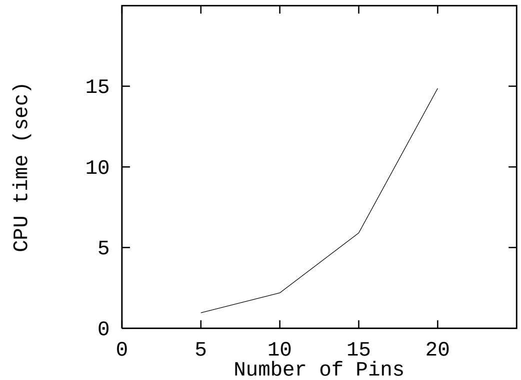

Figure 7: Variation of CPU times with pin counts.

the MVERT proposed here paper is useful. In real problems such situations are likely to occur often when routing trees have to be built to conform to timing budgets. This work also shows utility of considering nonHanan points as candidate Steiner points. The use of this procedure would reduce the congestion around the root node, particularly for technologies with low driver resistances where a SERT-like method would be likely to connect many nodes to the source, and also the overall congestion of the layout since the total wire length is reduced.

It is important to point out that the use of nonHanan nodes is valid only in scenario of Figure 1(b), and that the value of the timing constraints is important in determining whether nonHanan nodes will be exercised or not. As justified in Section 1, we believe that it is likely that in a real situation where the burden of delay reduction is shared by the gates and the interconnects, the timing constraints will lie within this range.

A primary result that follows from this paper is that it is not enough to consider points on the Hanan grid as candidate Steiner points. This work can be considered a first step in the direction of nonHanan routing. In real applications, this work must be extended to consider two more issues. Firstly, in its present form, this work does not address the issue of congestion and prerouted nets, and is therefore directly applicable to the most timing-critical global nets that are typically routed initially to allow the maximum routing flexibility for these nets. Just as efforts have been made to address the problem of simultaneous global routing of several nets using the Hanan grid (for example, [18]), similar efforts are needed in the nonHanan environment. Secondly, the role of buffer insertion is not considered here, but is important. By inserting a large enough number of buffers, the buffer resistance can be made to dominate interconnect resistance, and only minimum-length Steiner trees are required. However, this is not realistic since adding a large number of buffers can be expensive in terms of area, and may increase the delay of the net beyond the minimum. Therefore, the problem of optimal buffer insertion and topology construction in a nonHanan environment is an important problem to be addressed.

Acknowledgement: The authors thank J. Lillis for helping us realize that the nonHanan example in [2] had been published before our observations.

References

[1] K. D. Boese, A. B. Kahng, B. A. McCoy and G. Robins, "Fidelity and near-optimality of Elmore-based routing constructions," *Proc. IEEE Int. Conf. Computer Design*, pp. 81-84, 1993.

[2] K. D. Boese, A. B. Kahng, B. A. McCoy and G. Robins, "Rectilinear Steiner trees with minimum Elmore delay," *Proc. ACM/IEEE Design Automat. Conf.*, pp. 381-386, 1994.

[3] K. Boese, A. Kahng, B. McCoy, G. Robins, "Near-optimal critical sink routing tree constructions," *IEEE Trans. Computer-Aided Design*, Vol. 14, pp. 1417-1436, December 1995.

[4] J. Cong and C. K. Koh, "Interconnect layout optimization under higher-order RLC model," *Proc. IEEE Int. Conf. Computer Design*, pp. 713-719, 1997.

[5] J. Cong, K. S. Leung and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," *Proc. ACM/IEEE Design Automat. Conf.*, pp. 606-611, 1993.

[6] J. Cong, L. He, C. K. Koh and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration: The VLSI Journal*, Vol. 21, pp. 1-94, November 1996.

[7] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl.Math.*, Vol.14, pp. 255-265, 1966.

[8] N. Hasan, G. Vijayan, and C. K. Wong, "A neighborhood improvement algorithm for rectilinear Steiner trees," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 2869-2872, 1990.

[9] J. Ho, G. Vijayan and C. K. Wong., "Constructing the optimal rectilinear Steiner tree derivable from a minimum spanning tree," *Proc. IEEE Int. Conf. Computer Design*, pp. 6-9, 1989.

[10] J. Ho, G. Vijayan, and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem," *IEEE Trans. Computer-Aided Design*, Vol. 9, pp. 185-193, Feb.1990.

[11] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Applied Math.*, Vol.30, pp. 104-114, Jan.1976.

[12] F. K. Hwang, "An $O(n\log n)$ algorithm for suboptimal rectilinear Steiner tree," *IEEE Trans. Circuits Syst.*, Vol.CAS-26, pp. 75-77, Jan. 1979.

[13] A. Kahng and G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Trans. Computer-Aided Design*, Vol.11, pp. 893-902, July 1992.

[14] A. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, Norwell, MA, 1995.

[15] M. Kang, W. W.-M. Dai, T. Dillinger and D. LaPotin, "Delay bounded buffered tree construction for timing-driven floorplanning," *Proc. IEEE Int. Conf. Computer Design*, pp. 707-712, 1997.

[16] J. H. Lee, N. K. Bose, and F. K. Hwang, "Use of Steiner's problem in suboptimal routing in rectilinear metric," *IEEE Trans. Circuits Syst.*, Vol. CAS-23, pp. 470-476, July 1976.

[17] J. Lillis, C.-K. Cheng, T.-T. Y. Lin and C.-Y. Ho, "New performance-driven routing techniques with explicit area/delay tradeoffs and simultaneous wire sizing," *Proc. ACM/IEEE Design Automat. Conf.*, pp. 395-400, 1996.

[18] R. L. Carden IV, J. Li and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Computer-Aided Design*, pp. 208-216, Vol. 15, No. 2, February 1996.

[19] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design*, pp. 202-211, Vol. CAD-2, No. 3, July 1983.