Recent Developments in Performance Driven Steiner Routing:
an Overview

Manjit Borah  Robert Michael Owens  Mary Jane Irwin
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16801

Abstract
The contribution of interconnect delay to the stage delay of a circuit is increasing with scaling of the minimum feature size. At larger feature size the interconnect delay contribution was small and the driver resistance was very large compared to wire resistance. Consequently, a simple lumped model was sufficient for evaluating and optimizing circuit delay. However, with sub-micron processes, the contribution of interconnect delay dominates the stage delay and the wire resistance becomes noticeable, making the interconnect delay dependent on the routing topology. Hence it is becoming necessary to use a more accurate model for estimating and optimizing interconnect delay. This paper surveys the recent advancements in techniques for generating on-chip interconnect topology for optimizing circuit performance.

1 Introduction
As minimum feature size decreases and circuit size and complexity increases, the interconnect delay becomes the dominant component of circuit delay. The interconnect delay is the RC delay introduced due to the capacitance and the resistance of the metal wires connecting the nodes of the routing net. In earlier process technologies with long channel transistors, the intrinsic gate delays were significantly larger than the interconnect RC delay. Moreover, the driver ON resistance of a gate dominated over the wire resistance. Consequently, the circuit designers were more concerned about minimizing the gate delays and the total routing area and capacitance—the interconnect topology had little effect on the circuit performance.

As the feature sizes are scaled down, the gate delays improve with the scaling factor while local interconnect delay remain unchanged [2]. Hence, the delay contribution due to the interconnect becomes the dominant factor of the circuit delay in the sub-micron regime. Moreover, the increase in the wire resistance per unit length is quadratic with the scaling factor while the driver ON resistance is nearly unchanged [2]. Thus, the ratio of the driver ON resistance to the unit wire resistance is decreasing super-linearly with the scaling factor. The interconnect topology plays a significant role in the circuit delay. A simple lumped model for interconnect delay is no longer applicable. To complicate matters even more, the increasing circuit complexity is causing the global interconnect delays nearly independent of the driver strength due to resistive shielding effects[31, 16]. The delay of such interconnects must be optimized by inserting buffers or repeaters, using higher level metal layers and appropriately increasing the width of the wires. In this paper we consider only Steiner routing techniques for interconnect optimization; issues like buffer insertion and wire sizing need far more attention than the scope of this paper allows.

In a performance oriented physical design environment, placement, routing and other optimization steps such as transistor sizing and reordering are applied repeatedly in a feedback loop with the help of a timing analyzer providing the feedback information. Depending on the level of detail in the timing analysis, the feedback information may be the criticality of a net as a whole, identified critical sinks in the net, or a time bound given to each sink in the net. Based on the feedback information, different criteria may be used for routing. If a net is not considered critical (i.e., it does not belong to any critical path) then the length of the net may be minimized. If the feedback information for a critical net does not identify any sink as the critical sink, then the routing algorithm can either minimize the maximum delay to any sink (radius) of the net, or minimize the average delay of all the sinks. If the critical sink (i.e., the sink that lies on the critical path) is identified, then the delay between the source and the critical sink should be minimized. A more general case occurs when a time bound is assigned to each sink.

The rest of the paper is organized into the following
sections: section two discusses the timing model and intuitions derived from it. Section three surveys the techniques for minimizing interconnect capacitance for non-critical nets. Section four presents the techniques for optimizing interconnect delay when no critical sink information is known and section five presents the techniques for critical sink delay optimization. We conclude in section six with comments on future research directions.

2 Timing model

The delay of on-chip interconnect can be modeled as the distributed RC delay. A truly distributed RC model is difficult to compute accurately, therefore several lumped approximations have been proposed for estimating the RC delay quickly. Penfield et al, in [35] presented models to compute the upper and the lower bounds on the delay of an RC tree. The upper bound is computed as the sum of the open circuit time constants of all the nodes, a term which can be computed easily. Sakurai, in [36] also derived a simplified upper bound on the RC delay of a routing tree using empirical fit equations representing the contribution of the path resistance and the capacitance at a node.

A more accurate estimation is obtained using the Elmore delay [17]. The Elmore delay is computed as the 50% delay of the RC tree using the first (and the dominant) moment of its impulse response. The Elmore delay model provides an excellent tradeoff in terms of the computation time and accuracy. The Elmore delay from the source $n_0$ to sink $n_t$ is given by:

$$t_{n_0, n_t} = R_d \times C_0 + \sum_{e_i \in \text{path}(n_0, n_t)} r_i \times (c_i/2 + C_i)$$  \hspace{1cm} (1)

where,

- $n_0$: the source node of the routing tree
- $e_i$: the edge (wire) from node $n_i$ to its parent
- $r_i$: resistance of edge $e_i$
- $c_i$: capacitance of edge $e_i$
- $T_i$: the subtree rooted at node $n_i$
- $C_i$: the total capacitance of $T_i$ rooted at $n_i$
- $R_d$: driver ON resistance

Equation (1) uses a lumped approximation for each edge of the tree. Therefore the Elmore delay of an $n$-node tree can be computed in $O(n)$ time. A closer look at equation (1) reveals the following observations:

1. For on-chip interconnects, $R_d$ is typically much larger than the wire resistance, $r_i$. Hence, in order to minimize the first term in equation (1), the total tree capacitance $C_0$ should be minimized.

2. The resistance and capacitance of an edge (i.e., $r_i$ and $c_i$ of the edge $e_i$) are proportional to its length. Therefore, the second term of equation (1) is proportional to the square of the length of the path between $n_0$ and $n_t$. This quadratic dependency suggests that in order to minimize the Elmore delay for a given critical sink, the length of the path between the source and the critical sink should be minimized.

3. The capacitance of a subtree is multiplied by the resistance of the path between the source and the node of the subtree lying on the path between the source and the critical sink. Thus, the capacitance of a subtree near the sink has a larger multiplicative factor in the Elmore delay compared to a subtree near the source. Therefore subtrees with large capacitance should be close to the source.

The Elmore delay model is found to be a reliable model for characterizing the RC interconnect delay of a tree[4, 27]. Although the resistance shielding and transmission line effects in global interconnect lines are not captured accurately by the Elmore delay model, it provides a fast and reliable way to evaluate the signal delay for on-chip interconnects. Most of the recent techniques for interconnect optimization use the Elmore delay model for guiding the routing process.

3 Routing non-critical nets with minimum tree-length

When the net to be routed is not time critical, then we should minimize the total tree length. This has two implications. First, minimizing the tree length minimizes the routing area and hence the circuit area. Secondly, the capacitance of the net is minimized which reduces the switching capacitance of the node, reducing the power dissipation. A Steiner tree is used to minimize the interconnect length. Since optimal Steiner tree construction is an NP-hard problem [18], approximation heuristics are commonly used for generating Steiner routes. For a detailed study of Steiner routing heuristics the reader is referred to [26, 33].

Hwang [23] showed that any heuristic that improves upon the length of the minimum spanning tree (MST) generates a Steiner tree at most 1.5 times larger than the optimal Steiner tree. Encouraged by this result several heuristics have been developed that improves upon the minimum spanning tree. The simplest among the Steiner approximation heuristics is the edge-overlap class of heuristics [21, 28, 29]. We consider the heuristic presented in [21] as a representative of this class. This heuristic starts with an MST and merges the bounding boxes of
adjacent edges introducing Steiner points (figure 1). The resulting algorithm requires linear time on the number of nodes. Including the minimum spanning tree construction [24], it has $O(n \log n)$ complexity. Another class of heuristics [3, 34] emulates Kruskal's or Prim's minimum spanning tree construction [15]. Both of these classes of algorithms are fast, but produce routes far from optimal, only about 7 to 9% average reduction from the minimum spanning tree length.

![Figure 1: The edge-overlap heuristic](image)

To improve the route quality of Steiner approximations, Kahng and Robins, in [25] proposed a heuristic based on the 1-Steiner construction proposed by Georgakopoulos and Papadimitriou [19]. The basic algorithm repeatedly chooses the best Steiner point to add to the current set of points such that the length of the minimum spanning tree of the new set of points is minimized (figure 2).

![Figure 2: The 1-Steiner heuristic](image)

Using efficient methods to find the 1-Steiner point, they developed an $O(n^3)$ algorithm to compute a batched version of the 1-Steiner heuristic where several Steiner points are added in one pass. Using sophisticated techniques the asymptotic time complexity of batched 1-Steiner algorithm can be improved to $O(n^2 \log n)$ with $O(n^2 \log n)$ space requirement. The routes produced by the 1-Steiner heuristic is better than any approximation heuristic proposed so far. Salowe and Warne in [37] presented an algorithm for computing the optimal Steiner tree using a backtracking search technique. The algorithm required about half an hour to compute the optimal Steiner route for 30 points. A comparison with the optimal Steiner routes generated by [37] show that the batched 1-Steiner algorithm produces near optimal routes [20].

The 1-Steiner heuristic, however, is too slow for a reasonably large net. Since a routing algorithm has to be applied numerous times during the process of layout evaluation, a fast routing algorithm is desirable. Borah et al., in [8] presented a simple, yet efficient heuristic for computing the Steiner approximation from the minimum spanning tree (figure 3). This heuristic, known as the ER (edge-replacement) heuristic, repeatedly connects a node to the rectangular bounding box of a nearby edge, removing the longest edge in the cycle thus formed. A batched version of this algorithm is implemented using simple data-structures and techniques with $O(n^2)$ time complexity and $O(n)$ space. Using sophisticated techniques the running time of the batched ER heuristic can be further improved to $O(n \log n)$ [8]. The route quality of the batched ER heuristic is comparable to the batched 1-Steiner heuristic with one order speedup in running time.

Table 1 summarizes the results from the best existing heuristics for rectilinear Steiner trees. Columns 2 and 3 present the average reduction from the minimum spanning tree length by batched 1-Steiner and batched ER algorithm, respectively. Column 5 presents the improvements obtained by the optimal algorithm of [37]. It was noted in [8] that the routes produced by the 1-Steiner and the ER heuristics are different in nature and there are cases where one algorithm produces significantly better route than the other. Column 4 presents the results using a meta-heuristic where the BEST route of batched 1-Steiner and batched ER heuristic is considered. The running times of batched 1-Steiner, ER and the optimal algorithm are presented in columns 6, 7 and 8 respectively.

<table>
<thead>
<tr>
<th>net size</th>
<th>average improvement over MST</th>
<th>average CPU(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B1St</td>
<td>ER</td>
</tr>
<tr>
<td>4</td>
<td>8.54</td>
<td>8.54</td>
</tr>
<tr>
<td>5</td>
<td>9.34</td>
<td>9.34</td>
</tr>
<tr>
<td>6</td>
<td>9.79</td>
<td>9.78</td>
</tr>
<tr>
<td>8</td>
<td>10.06</td>
<td>10.07</td>
</tr>
<tr>
<td>10</td>
<td>10.19</td>
<td>10.16</td>
</tr>
<tr>
<td>14</td>
<td>10.32</td>
<td>10.28</td>
</tr>
<tr>
<td>20</td>
<td>10.51</td>
<td>10.47</td>
</tr>
<tr>
<td>25</td>
<td>10.47</td>
<td>10.40</td>
</tr>
<tr>
<td>30</td>
<td>10.48</td>
<td>10.41</td>
</tr>
</tbody>
</table>

139
4 Optimizing critical nets with no critical sink identified

When the timing analysis identifies a net as a whole as critical, but does not identify any particular sink on a critical path, then we may minimize either the maximum or the average delay of the net. The maximum delay of a net (radius of the net) is the maximum of the delays from the source to any of the sinks in the net. Using a geometric model, this problem is often formulated as the cost-radius tradeoff (figure 4). Minimizing the tree cost results in a Steiner topology and minimizing the radius leads to a shortest path tree (figure 4(b)).

![Cost-radius tradeoff](image)

(a) (b) (c)

Figure 4: The cost-radius tradeoff for a net with N sinks, N \( \gg 1 \) (source at the center): (a) minimum cost tree, cost = radius \( \approx R (1 + 2\pi) \); (b) minimum radius tree (shortest path tree) cost = \( N R \), radius = R; (c) with \( K \) spokes (\( N \gg K \)): cost \( \approx R (K + 2\pi) \), radius \( \approx R (1 + \frac{\pi}{K}) \).

Cohon et al, in [10] and Cong et al, in [11] first proposed approaches to minimize the radius of the routing tree using a geometric cost. The approach described in [10] uses a constrained tree construction around the geometric radius using a combination of the shortest path tree construction for minimum radius and the Steiner tree construction for minimum tree length. It starts with the main trunk of the tree connecting the two farthest nodes and connects the rest of the nodes minimizing the total length if the distance to the sink node from the source is within the geometric radius. Otherwise it connects the node using the shortest path from the source. The approach in [11] also minimizes the radius, recognizing the importance of minimizing the length of the routing tree. It allows the maximum radius of the tree to be \( 1 + \epsilon \) factor larger than the geometric radius. Hence, it allows a continuous tradeoff between the length of the routing tree and its radius using different values of \( \epsilon \).

Cong et al, in [12] proposed the “shallow-light” routing construct which bounds both, the radius and the total length of the tree simultaneously, allowing better control in the tradeoff. Alpert et al, in [1] showed that the objective of minimizing the radius and minimizing the total length can be achieved by combining the techniques of Prim’s and Dijkstra’s minimum spanning tree algorithms. In [13] the authors use an A-tree construction ¹ to obtain a balance between optimal Steiner tree and a shortest path tree using the intuitions drawn from the distributed RC model of [35].

While the above approaches for minimizing the radius use a linear delay model, recent approaches use the more accurate Elmore delay model. Boese et al, in [6] proposed method to minimize the maximum delay of a net using the Elmore delay model directly during the construction of the tree. The heuristic, known as the ERT (Elmore routing tree) heuristic, is based on Prim’s MST construction where at every step it connects to the growing tree, the node which minimizes the maximum Elmore delay of the current tree. The Steiner version of their algorithm (the SERT algorithm) has a complexity of \( O(n^4) \). In [5] they presented a branch-and-bound algorithm (BB-SORT) for computing the optimal Elmore routing tree for small (less than 10 nodes) nets. The authors of [5] also showed that the SERT algorithm produces routes with close to optimal Elmore delays (table 2, quoted from [5]). The extra wire-length overhead of the SERT route, compared to a minimum Steiner route is found to be very high.

**Table 2: Comparison of Elmore delay of SERT routes with optimal**

<table>
<thead>
<tr>
<th>net size</th>
<th>% above opt Elmore delay</th>
<th>SERT CPU(s)</th>
<th>BB-SORT CPU(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.3</td>
<td>0.0014</td>
<td>0.015</td>
</tr>
<tr>
<td>6</td>
<td>1.6</td>
<td>0.003</td>
<td>0.13</td>
</tr>
<tr>
<td>7</td>
<td>2.4</td>
<td>0.0056</td>
<td>1.3</td>
</tr>
<tr>
<td>8</td>
<td>3.2</td>
<td>0.01</td>
<td>14.4</td>
</tr>
<tr>
<td>9</td>
<td>3.9</td>
<td>0.016</td>
<td>61.8</td>
</tr>
</tbody>
</table>

All of the above critical net methods minimize the maximum delay to any sink in the net. Recently, Vittal et al, in [38] developed an algorithm to compute routing trees to minimize the average Elmore delay of a net using alphabetic trees with \( O(n^2) \) time complexity. Their approach is based on a global strategy that bounds the total tree length using the inherent property of the alphabetic tree data-structure and uses the weight balancing technique to minimize the average Elmore delay to all the sinks. However, the results of this technique is sensitive to the alphabetic order chosen for the initial tree. Although the formulation of the problem in [38] is different from that of [6] the authors compared their results with [6] and showed that both algorithms produce routes with comparable average delays. The area (wire-length) overhead of the alphabetic tree is usually higher than SERT.

¹ A rectilinear Steiner tree where every path from the source to any sink is a shortest path.
5 Optimizing interconnects with identified critical sinks

If the feedback information to the routing algorithm identifies the critical paths in the circuit (i.e., identifies the critical sink in a net), then the routing heuristic should concentrate on minimizing the interconnect delay to the identified critical sink. Earliest attempt at routing to minimize sink delay is found in [32], where the authors use an A* algorithm to compute the route with maximized minimum delay slack to any sink. This is a generalized version of performance driven routing where a time bound is computed for each sink using a timing analyzer and the timing slack at each pin is maximized. The delay model used in this algorithm is that of [36] which is less accurate than the Elmore delay model.

In [30] Lim, Cheng and Wu gave an algorithm for minimizing the delay to identified critical sinks (POMRSTST algorithm) using a geometric cost function in which they minimize a combination of the total tree length and the source-to-sink distance, based on Prim’s MST construction. Their algorithm has \(O(n^3)\) time complexity. Hong et al [22] also proposed timing driven routing based on the simplified wire delay model of [30] as the upper bound on the delay. They developed two algorithms, one derived from Dreyfus-Wagner’s dynamic programming method and the other, the Constructive Force Directed algorithm, based on Kruskal’s minimum spanning tree (MST) algorithm. They expressed the critical sink delay as a function of the total tree-length and the source to sink delay. The algorithm based on Dreyfus-Wagner’s method has exponential time complexity which makes it impractical for large nets. The CFD algorithm has asymptotic complexity \(O(n^2)\).

Boese et al, in [6] developed a routing heuristic for minimizing the Elmore delay to the identified critical sink (the SERT-C algorithm). They first connect the critical sink to the source with minimum length wire and then grow the Steiner tree for the rest of the sinks in a manner similar to Prim’s algorithm, using the current Elmore delay to the critical sink as the cost criterion to be minimized. The algorithm has an \(O(n^2)\) time complexity where \(n\) is the number of nodes in the net. They also developed a branch-and-bound algorithm to compute the routing tree with optimal Elmore delay to the identified critical sink (the BBSORT-C algorithm)[5] which can handle small nets (less than 10 nodes) in reasonable time.

The SERT-C heuristic often produces routes with critical sink delay far from optimal. This is due to the lack of global information regarding sub-tree capacitance during the construction of the tree. For example, when connecting a new node to the growing tree, the current Elmore delay to the critical sink is considered for minimization. This cost function is oblivious to the capacitance of the sub-tree that may be connected to the node in the final route. Thus the final routing tree produced by SERT-C often contains large capacitive sub-trees connected near the critical sink, resulting in large delays to the sink (Figure 5(a)).

Figure 5: Critical sink routing for a 10-terminal net: (a) SERT-C route, critical sink Elmore delay=1.0 nS; (b) PER Steiner route, critical sink Elmore delay=0.72 nS

Borah et al, in [9] presented an algorithm based on their edge-based (ER) heuristic [8] to compute Steiner routes with minimized Elmore delay to the critical sink (Figure 5(b)). Their algorithm (the PER-Steiner algorithm) uses three phases, each phase using the same ER heuristic with a different cost function. The three phases concentrate on (1) the total tree length, (2) the source-sink distance and (3) moving capacitive sub-trees towards the source, respectively. The cost functions in phase (2) and (3) use the Elmore delay model directly to determine the incremental effect of an ER operation on the critical sink delay [9]. The PER-Steiner algorithm has \(O(n^2)\) running time. A direct comparison between the PER-Steiner algorithm and the SERT-C algorithm showed that the PER-Steiner algorithm produces routes with significantly better Elmore delay to the critical sink than SERT-C (Table 3) with similar CPU time requirements [7]. Comparison with the optimal critical sink

Table 3: Comparison of average delay and running time of PER-Steiner and SERT-C (200 samples each)

<table>
<thead>
<tr>
<th>Size of net</th>
<th>avg critical sink delay SERT-C/PER-Steiner</th>
<th>PER-Steiner time (sec)</th>
<th>SERT-C time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1.10</td>
<td>0.014</td>
<td>0.002</td>
</tr>
<tr>
<td>20</td>
<td>1.19</td>
<td>0.06</td>
<td>0.01</td>
</tr>
<tr>
<td>50</td>
<td>1.37</td>
<td>2.5</td>
<td>0.07</td>
</tr>
<tr>
<td>100</td>
<td>1.68</td>
<td>2.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Technology: \(r = 0.03\Omega/\mu\text{m}, c = 3.52e - 16 F/\mu\text{m}, R_d = 100.0\Omega\) and \(c_{\text{no,load}} = 1.53e - 14 F\)
Table 4: Comparison of critical sink Elmore delay with optimal delay

<table>
<thead>
<tr>
<th>Size of net</th>
<th>percent deviation from optimal Elmore delay (PER-Stz vs. SERT-C)</th>
<th>average CPU time (100 samples) PER-Stz vs. SERT-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.52 4.11</td>
<td>3.7 ms 0.5 ms</td>
</tr>
<tr>
<td>5</td>
<td>1.09 5.10</td>
<td>4.2 ms 0.6 ms</td>
</tr>
<tr>
<td>6</td>
<td>1.43 8.7</td>
<td>6.0 ms 0.9 ms</td>
</tr>
<tr>
<td>7</td>
<td>1.55 12.37</td>
<td>7.5 ms 1.3 ms</td>
</tr>
<tr>
<td>8</td>
<td>1.85 14.13</td>
<td>8.2 ms 1.5 ms</td>
</tr>
<tr>
<td>9</td>
<td>2.05 16.06</td>
<td>8.8 ms 1.9 ms</td>
</tr>
</tbody>
</table>

routes produced by the BBSORT-C algorithm [5] showed that the PER-Steiner algorithm on the average produces routes with nearly optimal Elmore delay to the critical sink (table 4).

6 Conclusions

A survey of the recent techniques for performance oriented Steiner routing is presented. Three cases of routing are identified, namely, routing non-critical nets with minimum wire-length, routing critical nets without identified sinks and routing critical nets with identified critical sinks. In all the three cases, practical algorithms have been reported which produce close to optimal approximations. The general case of critical net routing where a time bound is specified for each sink in the net may become important as dominance of interconnect delay increases and the interconnect delay analysis techniques mature. Performance driven routing of nets with multiple sources (such as buses) has also been considered [14].

References