Dynamically-Wiresized Elmore-Based Routing Constructions*

Todd D. Hodes, Bernard A. McCoy and Gabriel Robins

Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442

Abstract

We analyze the impact of wiresizing on the performance of Elmore-based routing constructions. Whereas previous wiresizing schemes are *static* (i.e., they wiresize an existing topology), we introduce a new *dynamic* Elmore-based wiresizing technique, which uses wiresizing considerations to *drive* the routing construction itself. Simulations show that dynamic wiresizing affords superior performance over static wiresizing, and also avoids topological degeneracies. Moreover, dynamically-wiresized Elmore-based routing constructions significantly outperform all previous methods in term of maximum source-sink signal delay, affording up to 77% SPICE delay improvement over traditional Steiner routing.

1 Introduction

Interconnect delay has recently become a dominant concern in the design of complex, high-performance circuits, due to the scaling of VLSI technology [9] [36]. Performance-driven layout design has thus become an active area of research over the past several years, where for a given signal net, the typical goal of performance-driven routing is to minimize average or maximum source-sink delay [23]. Much early work implicitly equates optimal routing with minimum-cost Steiner routing. For example, Dunlop et al. [10] use static timing analysis to yield net priorities, so that the highest-priority nets may be routed by minimum Steiner trees, leaving lower-priority nets to subsequently encounter blockages. Jackson, Kuh, and Marek-Sadowska [21] and Prasitjutrakul and Kubitz [29] have given approaches which are tuned to building-block layout and allow prescribed upper bounds on individual source-sink delays.

*Corresponding author is Professor Gabriel Robins, Department of Computer Science, Thornton Hall, University of Virginia, Charlottesville, VA 22903-2442, Email: robins@cs.virginia.edu, phone: (804) 982-2207, FAX: (804) 982-2214.

Recently it became increasingly apparent that for leading-edge technologies, delay minimization and wirelength minimization are far from synonymous. For example, Cohoon and Randall [6] proposed a heuristic which simultaneously considered both the *cost* (total edge length) and the *radius* (longest source-sink path length) of the routing tree. A more general formulation was given by Cong et al. [7], wherein a parameter ϵ guides the tradeoff between cost and radius minimization in a "provably good" BRBC (bounded-radius, bounded-cost) algorithm, which affords both cost and radius simultaneously within *constant* factors of optimal. The cost-radius tradeoff may also be viewed as one between competing minimum spanning tree (MST) (or minimum-cost Steiner tree) and shortest-path tree (SPT) constructions. Using this perspective, Alpert et al. [1] recently proposed the AHHK algorithm, which achieves a direct MST-SPT tradeoff. Finally, Cong et al. [8] have recently proposed the use of rectilinear Steiner arborescences [31], or A-Trees, which are essentially minimum-cost SPTs with Steiner points allowed. As noted in [8], *wiresizing* can significantly improve signal delay in a given routing.

There are two common shortcomings to previous high-performance routing methods: (1) their optimization criteria are primarily "geometric" in nature (as opposed to minimizing physical delay), and (2) they are "oblivious" to particular technology parameters (i.e., they produce the *same* routing construction for *different* values of wire resistance, capacitance, etc.). To overcome these flaws, Boese et al. [5] have recently developed a construction which greedily optimizes the Elmore delay formula *directly* to produce low-delay routing trees. Not only are these constructions adaptable to the prevailing technology parameters, but they were found to be near-optimal with respect to Elmore delay for a wide range of technology parameters [3]. Moreover, it was shown that Elmore delay has high fidelity to physical (SPICE-computed) delay over a range of IC technologies, i.e. near-optimal Elmore delay implies near-optimal SPICE delay [4] and [3] [25]. One drwaback of the methods of [5] is that they sometimes produce degenerate routings (i.e., star-like topologies).

In this paper, we analyze the impact of wiresizing on the performance of Elmore-based routing constructions. Whereas previous wiresizing schemes are *static* (i.e., they take as input a complete fixed routing topology and then find a good wiresizing for it), we introduce a new practical Elmore-based wiresizing technique that is *dynamic* (i.e., we use wiresizing considerations to *drive* the routing construction itself). Our empirical data shows that dynamic wiresizing affords superior performance over static wiresizing, and also avoids degenerate star-like topologies. Moreover, we show that dynamically-wiresized Elmore-based routing constructions outperform all previous methods, yielding up to 77% reduction in SPICE delay over traditional Steiner routing.

The remainder of this paper is organized as follows. Section 2 gives basic definitions, formalizes the problem of constructing optimal-delay interconnection topologies, and discusses the delay models. In Section 3 we review three of the best known routing constructions. Section 4 discusses the *static* greedy wiresizing algorithm. In Section 5 we develop our new heuristic which combines the low-delay routing and dynamic wiresizing methods. Section 6 presents experimental results, and we conclude in Section 7. This work is to appear in [17].

2 Problem Formulation

Our overall goal is as follows: given an arbitrary set of pins with a designated source, we wish to electrically connect all the pins so that the maximum source-sink signal propagation delay is minimized. Ideally, a routing algorithm will compute and optimize signal delays according to a detailed circuit simulation, such as that provided by SPICE [28]. However, the computation times required by SPICE are prohibitive for routing tree construction, and therefore more efficient delay estimators are needed. As recently shown by Boese et al. [3] both the *fidelity* and *accuracy* of Elmore's distributed RC delay approximation is surprisingly high with respect to more complex delay estimators, such as the "Two-Pole" distributed RCL simulator of [38], as well as the SPICE circuit simulator [28]. We therefore use the Elmore formula to compare our routing constructions to existing ones.

We begin with some definitions and notation. A *signal net* $N = \{n_0, n_1, ..., n_k\}$ is a fixed set of *pins* in the Manhattan plane to be connected by a *routing graph* $G = (N, E)$, where $E \subseteq N \times N$. Pin $n_0 \in N$ is a *source* (i.e., where the signal originates), and the remaining pins are *sinks* (i.e., where the signal propagates to). Each edge $e_{ij} \in E$ has an associated *edge cost*, d_{ij}, equal to the Manhattan distance between its two endpoints n_i and n_j; the *cost* of G is the sum of its edge costs. We use $t(n_i)$ to denote the signal propagation delay from the source to pin n_i. Our goal is to construct a routing which spans the net and which minimizes the maximum source-sink delay:

Optimal Steiner Routing Tree (OSRT) Problem: Given a signal net $N = \{n_0, n_1, ..., n_k\}$ with source n_0, find a set of points S and construct a routing tree $T = (N \cup S, E)$, $E \subseteq N \cup S \times N \cup S$, such that $t(T) = \max_{i=1}^{k} t(n_i)$ is minimized.

Elmore delay [11] [34] is defined as follows. Given a routing tree T rooted at n_0, let e_i

denote the edge from n_i to its parent. The resistance and capacitance of edge e_i are denoted by r_{e_i} and c_{e_i}, respectively. Let T_i denote the subtree of T rooted at n_i, and let c_i denote the sink capacitance of n_i. We use C_i to denote the *tree capacitance* of T_i, namely the sum of sink and edge capacitances in T_i. Using this notation, the Elmore delay along edge e_i is equal to $r_{e_i}(c_{e_i}/2 + C_i)$. Let r_d denote the output driver resistance at the net's source. The Elmore delay $t_{ED}(n_i)$ at sink n_i is:

$$t_{ED}(n_i) = r_d C_{n_0} \quad + \sum_{e_j \in path(n_0,n_i)} r_{e_j}(c_{e_j}/2 + C_j) \tag{1}$$

Elmore delay has a compact definition and can be quickly evaluated at *all* sinks in $O(k)$ time [34]. The calculation uses two depth-first traversals: (1) to compute the delay along each edge and (2) to sum up the delays along each source-sink path; this enables an efficient implementation.

3 Three Routing Constructions

Before be present our new dynamically-wiresized Elmore-based routing construction, we first review three of the best existing routing methods: (1) Iterated 1-Steiner, (2) Elmore Routing Tree, and (3) A-Tree.

3.1 The Iterated 1-Steiner Construction

For a given set P of n points in the plane, an *edge* (i.e., wire) between two points $x \in P$ and $y \in P$ is denoted by (x, y). The *cost* of an edge is the Rectilinear (i.e., Manhattan) distance between its endpoints. A *spanning tree* over P is a set T of $n - 1$ edges with endpoints in P such that the induced graph is connected. The *cost* of a tree T, denoted $\overline{\mathrm{T}}$, is the sum of the costs of its edges. A *minimum spanning tree* (MST) is a spanning tree having least cost. Thus we denote the minimum spanning tree itself by MST and the *cost* of the minimum spanning tree by $\overline{\mathrm{MST}}$. A Steiner tree is a spanning tree over the original pointset P and a (possibly empty) additional pointset S (i.e., the *Steiner* points). We are now ready to define the minimum rectilinear Steiner tree problem:

The Minimum Rectilinear Steiner Tree (MRST) problem: Given a set P of n points in the Manhattan plane, find a set S of *Steiner points* such that the minimum spanning tree (MST) over $P \cup S$ has minimum cost.

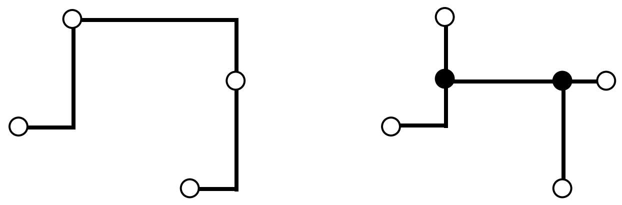

Figure 1: A minimum spanning tree (left) and MRST (right) for a fixed net.

Figure 1 shows an MST and an MRST for a fixed pointset. As with the MST, we denote the Steiner tree itself by MRST and the cost of this tree as $\overline{\text{MRST}}$. Research on the MRST problem has been guided by several fundamental results. First, Hanan [14] has shown that there always exists an MRST with Steiner points chosen from the intersection of all the horizontal and vertical lines passing through the points in P (see Figure 2); indeed this result generalizes to all higher dimensions [35]. However, a second major result establishes that despite this restriction on the solution space, the MRST problem remains NP-complete [12], prompting a large number of heuristics, as surveyed in [20].

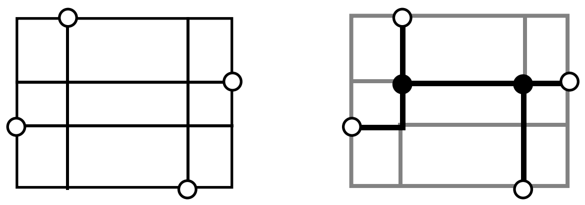

Figure 2: Hanan's theorem: there always exists an MRST with Steiner points chosen from the intersection of all the horizontal and vertical lines passing through all the points.

In solving intractable problems, we often seek provably good heuristics having bounded worst-case error from optimal. Thus, a third important result establishes that the rectilinear MST is a fairly good approximation to the MRST, with a worst-case performance ratio of $\overline{\text{MST}}/\overline{\text{MRST}} \leq \frac{3}{2}$ [18]. This implies that any MST-based strategy which improves upon an initial MST topology will also enjoy a performance ratio of at most $\frac{3}{2}$, which has prompted a large number of Steiner tree heuristics that resemble classic MST construction methods [15] [16] [19] [26] [27], all producing Steiner trees with average cost 7% to 9% smaller than MST cost [32] [37].

Unfortunately, all MST-based MRST constructions were recently shown to have a worst-case performance ratio of exactly $\frac{3}{2}$ [24]. This negative result has motivated research into alternate schemes for MRST approximation, with the best performing among these being the Iterated 1-Steiner (I1S) algorithm [23] [22]. I1S always performs strictly better than $\frac{3}{2}$ times optimal [33], and achieves almost 11% average improvement over MST cost. It was shown in [2] that for typical nets, I1S has average performance of less than 0.25% from optimal and produces optimal solutions up to 90% of the time. For two pointsets P and S, define the MST savings of S with respect to P as:

$$\Delta\overline{\mathrm{MST}}(P,S) = \overline{\mathrm{MST}}(P) - \overline{\mathrm{MST}}(P \cup S)$$

We use H(P) to denote the set of Hanan Steiner point candidates (i.e., the intersections of all horizontal and vertical lines passing through points of P). For a pointset P, a *1-Steiner point* $x \in H(P)$ maximizes $\Delta\overline{\mathrm{MST}}(P,\{x\}) > 0$. The I1S method repeatedly finds 1-Steiner points and includes them into S. The cost of the MST over $P \cup S$ will decrease with each added point, and the construction terminates when there is no x with $\Delta\overline{\mathrm{MST}}(P \cup S,\{x\}) > 0$. Although a Steiner tree may contain at most $n-2$ Steiner points [13], I1S may add more than $n-2$ Steiner points; therefore, at each step we eliminate any extraneous Steiner points having degree 2 or less in the MST. Figure 3 illustrates a sample execution of I1S, and Figure 4 describes the algorithm formally.

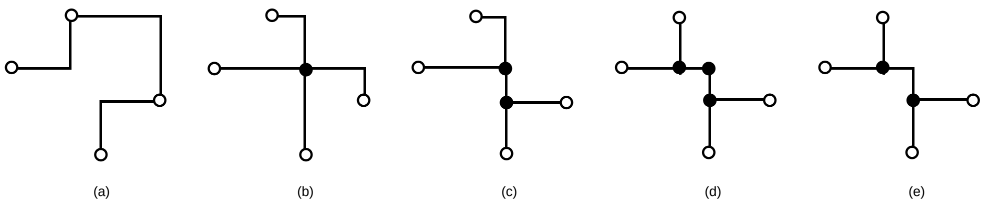

Figure 3: Execution of Iterated 1-Steiner (I1S) on a 4-point example. Note that in step (d) a degree-2 Steiner point is formed and is thus eliminated from the topology (e).

3.2 The Elmore Routing Tree (ERT) Construction

While it is known that delay in a routing tree is a non-linear phenomenon [11], many previous methods for routing tree construction have either implicitly or explicitly assumed that delay is proportional to source-sink path length. Thus, such methods only attempt to *heuristically* capture the goal of "high performance," and it is therefore not surprising that when trees produced by these methods were tested by simulation, their performance often proved disappointing. The SERT construction avoids the level of abstraction inherent in such previous

```
The Iterated 1-Steiner (I1S) Algorithm
Input: A set $P$ of $n$ points
Output: A rectilinear Steiner tree which spans $P$
$S = \emptyset$
While $T = \{x \in H(P) | \Delta\overline{\mathrm{MST}}(P \cup S, \{x\}) > 0\} \neq \emptyset$ Do
    Find $x \in T$ with maximum $\Delta\overline{\mathrm{MST}}(P \cup S, \{x\})$
    $S = S \cup \{x\}$
    Remove from $S$ points with degree $\leq 2$ in $\mathrm{MST}(P \cup S)$
Output $\mathrm{MST}(P \cup S)$
```

Figure 4: Algorithm I1S: a near-optimal spanning tree.

objectives as "minimum cost" or "bounded radius" and instead *directly* optimizes Elmore delay in the tree construction.

The SERT algorithm is analogous to Prim's minimum spanning tree construction [30], and works as follows. Starting with the trivial tree consisting initially of only the source pin, we grow the tree at each step by finding a new pin to connect to the tree, so that the maximum Elmore delay to any leaf is minimized; but rather than restricting the new pin to connect directly to a pin already in the tree, we allow the new pin to connect to some tree edge, thus inducing a Steiner point. In other words, given a tree $T = (V, E)$, we iteratively find $u \notin V$, $(v, v') \in E$, and a new point w on edge (v, v') to minimize the maximum source-sink Elmore delay in the tree $(V \cup \{u, w\}, (E - \{(v, v')\}) \cup \{(v, w), (w, v'), (u, w)\})$. We then add u and w to V, and replace E by $(E - \{(v, v')\}) \cup \{(v, w), (w, v'), (u, w)\}$. Again, the algorithm terminates when the resulting Steiner tree spans the entire net. A formal description of the algorithm, called the *Steiner Elmore routing tree* (SERT) construction, is given in Figure 5.

```
The Steiner Elmore Routing Tree (SERT) Algorithm
Input: A signal net $N$ with a source $n_0 \in N$
Output: A low-delay Steiner tree which spans N
  $T = (V, E) = (\{n_0\}, \emptyset)$
  $M = N - \{n_0\}$
  While $M \neq \emptyset$ do
      Find $u \in M$, $(v, v') \in E$, and a new point $w$ which minimizes the maximum Elmore delay
              from $n_0$ to any leaf in the tree $(V \cup \{u, w\}, (E - \{(v, v')\}) \cup \{(v, w), (w, v')\})$
      $V = V \cup \{u, w\}$
      $E = (E - \{(v, v')\}) \cup \{(v, w)\} \cup \{(w, v')\})$
      $M = M - \{u\}$
  Output resulting Steiner tree $T = (V, E)$
```

Figure 5: Algorithm SERT: constructing a low-delay Steiner Elmore routing tree for a given net.

3.3 The A-Tree Construction

The A-Tree has been used by [8] as the preferred interconnect topology because it minimizes the wirelength while maintaining shortest paths between the source and every sink. An A-Tree can be generated using the generalized rectilinear Steiner arborescence [31]. The algorithm begins with a forest of n single-node arborescences and proceeds by applying a sequence of either "optimal" or "heuristic" moves that extends an existing arborescence or combines two arborescences; the process terminates when only one arborescence remains (see [8] for more details). The A-Tree algorithm is formalized in Figure 6.

```
The A-Tree Algorithm
Input: A signal net $N$ with source $n_0 \in N$
Output: A routing tree $T$ which spans $N$
  $T = (V, E) = (\{n_0\}, \emptyset)$
  While there is more than one arborescence Do
      If $\exists$ optimal moves Then perform an optimal move
      Else perform a heuristic move
  Output resulting Steiner tree $T = (V, E)$
```

Figure 6: Algorithm A-Tree: a generalization of the rectilinear Steiner arborescence.

4 Static Wiresizing

Wiresizing (i.e., increasing the widths of certain wires) can improve signal propagation delay by trading-off capacitance for resistance: when a wire width is increased, additional capacitance is induced, but overall source-sink resistance may decrease. The idea behind wiresizing is to find wire segments in the routing where an increase in capacitance is more than compensated for by the corresponding decrease in resistance, thus improving the maximum source-sink routing signal delay. Given a fixed tree T, let $w(e_i)$ denote the width assignment of edge e_i and for simplicity we let $w(e_i)$ range over a discreet set of values $\{w_1, w_2, ..., w_k\}$. We are now ready to extend our problem of OSRT to include wiresizing:

Optimal Wiresized Steiner Routing Tree (OWSRT) Problem: Given a signal net $N = \{n_1, n_2, ..., n_k\}$ with source n_0 and a set of widths $W = \{w_0, w_1, ..., w_j\}$ where $w_0 < w_1 < \cdots < w_j$, find a set of points S and construct a routing graph $T = (N \cup S, E)$, $E \subseteq N \cup S \times N \cup S$, such that for each $e \in E$ with $w(e) \in W$, $t(T) = \max_{i=1}^{k} t(n_i)$ is minimized.

Given a fixed topology, the greedy wiresizing scheme of [8] recursively wiresizes each subtree of the source; as long as overall maximum tree delay improvement is possible, each edge connecting the root to a subtree is widened. This *static greedy wiresizing* (SGW) scheme is formalized in Figure 7; it generalizes the greedy wiresizing scheme of [8], in that it allows for an arbitrary delay calculation to be used. Note that this method is *static*, meaning that the topology of the tree is determined *before* wiresizing commences and does not change during the wiresizing process.

```
The Static Greedy Wiresizing (SGW) Algorithm
Input: A tree $T = (V, E)$ with source $n_0 \in N$ and a set $W$ of edge widths
Output: A wiresized tree $T_w$ which spans $N$
  For each node $n_i \in V$ such that $e = (n_0, n_i) \in E$ Do
    Call SGW on the subtree routed at $n_i$
      Repeat
        delay$_{old}$ = $t(T)$
        Increase $w_r$ to $w_{r+1}$ of edge $e$
    Until delay$_{old} < t(T)$
    Decrease $w_r$ to $w_{r-1}$ of edge $e$
```

Figure 7: Algorithm SGW: the static greedy wiresizing algorithm.

5 A New Dynamic Wiresizing Construction

While static greedy wiresizing provides a near-optimal wiresizing for a given topology [8], the wiresizing process is largely constrained by that fixed input topology. Ideally we would like to compute the *optimal* combination of routing topology *and* wiresizing; unfortunately, this is not computationally feasible. On the other hand, we do not want to completely dissociate the topology construction from the wiresizing issues (as was done in [8]), since such a strategy will not benefit from a possible synergy between these two issues.

With this in mind, we have developed a *dynamic* wiresizing algorithm that hybridizes the routing topology construction with the wiresizing process. Our new construction combines the Elmore routing tree method of Section 3.2 with the greedy wiresizing of Section 4. The overall structure of the *dynamically wiresized Steiner Elmore routing tree* (DWSERT) construction is similar to that of SERT, except that when we select a new edge to add to the growing topology, instead of minimizing the Elmore delay in the resulting topology, we seek to minimize the Elmore delay in the *wiresized* current topology; in other words, in each step of the SERT construction we invoke the SGW routine once for each candidate edge and add the edge that yields the best wiresized tree.

Note however, that during the execution of DWSERT, a partial topology is not *actually* wiresized, but instead its edges are left having the minimum width; rather, wiresizing considerations are used as a *guide* to drive the edge-selection process. When the topology spans all the net pins, we invoke the static wiresizing algorithm one final time and return the resulting *wiresized* tree. The DWSERT algorithm is formalized in Figure 8.

```
The Dynamically Wiresized Steiner Elmore Routing Tree (DWSERT) Algorithm
Input: A signal net $N$ with source $n_0 \in N$
Output: A wiresized low-delay Steiner tree which spans N
  $T = (V, E) = (\{n_0\}, \emptyset)$
  $M = N - \{n_0\}$
  While $M \neq \emptyset$ do
      Find $u \in M$, $(q, q') \in E$, and a new point $p$ which minimizes the maximum Elmore delay
          from $n_0$ to any leaf in the wiresized tree SGW($V \cup \{u, p\}, (E - \{(q, q')\}) \cup \{(q, p), (p, q')\}$
      $V = V \cup \{u, w\}$
      $E = (E - \{(q, q')\}) \cup \{(q, p)\} \cup \{(p, q')\})$
      $M = M - \{u\}$
  Output SGW($T = (V, E)$)
```

Figure 8: Algorithm DWSERT: constructing a *dynamically* wiresized low-delay Steiner tree for a given net.

6 Experimental Results

We have implemented the I1S, SERT, and the A-Tree algorithms, their statically wiresized versions (WI1S, WSERT, and WA-Tree, respectively), and DWSERT, the dynamically wiresized version of SERT, using C in the UNIX Sun environment. The code is available upon request. We tested these algorithms on random nets of 5, 10, 15, 20, 25, and 30 pins, uniformly distributed in the $100000\mu \times 100000\mu$ grid, with the source being one of the pins chosen at random. Our technology parameters correspond to a typical MCM technology, and are summarized in Table 1.

Table 2 gives the average percent improvement in maximum source-sink delay as a percentage over the corresponding I1S values. In other words, each entry in the table represents the percentage improvement of the maximum delay as compared to the maximum delay for the I1S routing over the same net. The results are averaged over 50 random pointsets. We see from Table 2 that static wiresizing decreases average maximum delay dramatically when it is applied to either an I1S tree or an A-Tree: WI1S has a maximum delay as much as 38% less than I1S, while WA-Tree exhibits a delay improvement of as much as 15% less than I1S. We also see that the benefit of dynamic wiresizing increases with the net size. Figure 9

Parameter	**Value**
driver resistance	25 Ω
wire resistance	0.008 $\Omega/\mu m$
wire capacitance	0.060 $fF/\mu m$
wire inductance	380 $fH/\mu m$
sink loading capacitance	1000 fF
layout area	10^4 mm^2

Table 1: These multichip module interconnect parameters were provided by the AT&T Microelectronics Division.

summarizes the data pictorially.

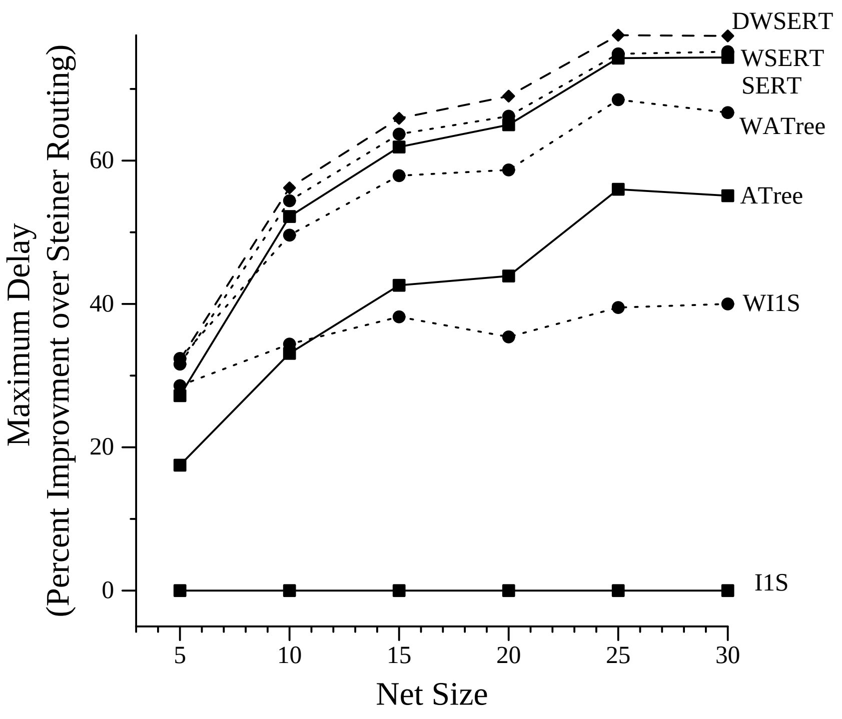

Figure 9: SPICE simulation results comparing the Iterated 1-Steiner, SERT, and A-Tree constructions, as well as their wiresized versions. Results are normalized to 1-Steiner. Simulations are over the MCM technology for 50 randomly distributed nets with uniform distribution.

In contrast, very little improvement occurs when an SERT is statically wiresized. This is because near-optimal MCM SERT topologies are star-like, each sink being directly connected to the source and having a relatively large loading capacitance; thus the lower resistance of a wider edge can not overcome the higher overall capacitance. On the other hand, DWSERT does not yield such degenerate topologies, which is another advantage of the dynamic wiresizing method. We also see that both the WI1S tree and WA-Tree still perform worse on average than a *non-wiresized* SERT. DWSERT improves over WA-Tree by up to 10%, and is thus the clear winner among the various methods. Figure 10 depicts the wiresized Iterated 1-Steiner, A-Tree, and SERT constructions for the same random 20-pin net.

	Elmore Delay					
	$\|N\| = 5$	$\|N\| = 10$	$\|N\| = 15$	$\|N\| = 20$	$\|N\| = 25$	$\|N\| = 30$
I1S	0.0	0.0	0.0	0.0	0.0	0.0
WI1S	42.1	51.9	55.1	58.0	55.9	56.9
SERT	33.8	53.8	62.8	68.0	67.8	71.0
WSERT	39.2	56.3	64.2	69.2	69.3	71.8
DWSERT	46.8	61.3	67.5	71.8	71.8	73.9
ATree	17.6	34.0	44.4	48.5	46.3	50.2
WA-Tree	40.6	58.6	63.9	63.9	67.9	68.7

	SPICE Delay					
	$\|N\| = 5$	$\|N\| = 10$	$\|N\| = 15$	$\|N\| = 20$	$\|N\| = 25$	$\|N\| = 30$
I1S	0.0	0.0	0.0	0.0	0.0	0.0
I1SWS	28.6	34.4	38.2	35.4	39.5	40.0
SERT	27.2	52.2	61.9	65.0	74.3	74.4
WSERT	31.6	54.4	63.7	66.2	74.9	75.2
DWSERT	32.2	56.2	65.9	69.0	77.5	77.4
ATree	17.5	33.1	42.6	43.9	56.0	55.1
WA-Tree	32.4	49.6	57.9	58.7	68.5	66.7

Table 2: Elmore and SPICE simulation results comparing the Iterated 1-Steiner, SERT, and A-Tree constructions, as well as their wiresized versions. Each entry corresponds to an average percent improvement over Iterated 1-Steiner delay. 50 random (uniformly distributed) nets were used per each net size. Both Elmore (top) and SPICE (bottom) were used to compute signal delays.

7 Conclusions

We have analyzed the impact of wiresizing on the performance of Elmore-based routing constructions. Previous wiresizing schemes are *static* (i.e., they wiresize a fixed existing topology); in contrast, we introduced a new *dynamic* Elmore-based wiresizing technique, using wiresizing considerations to *drive* the routing construction itself. Simulations indicate that dynamic wiresizing affords improved performance over static wiresizing, and yield more favorable (i.e., non-star) topologies. Moreover, dynamically-wiresized ELmore-based constructions seem to significantly outperform all previous methods in term of maximum source-sink SPICE delay, affording up to 77% delay improvement over traditional Steiner routing, as measured by SPICE.

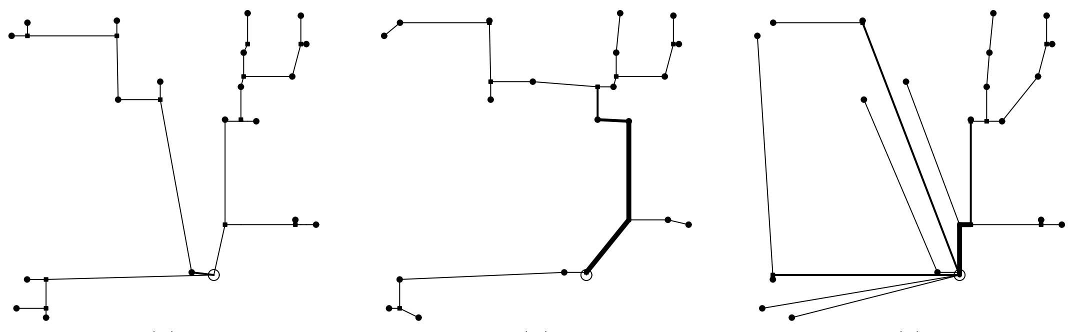

Figure 10: A comparison of the different constructions for a random 20-pin net: (a) the statically Wiresized A-Tree has maximum source-sink delay of 10.16ns (the non-wiresized A-Tree has a delay of 10.42ns; (b) the (statically) wiresized Iterated 1-Steiner tree has delay of 9.53 ns (the non-wiresized Iterated 1-Steiner tree has delay of 22.06ns); (c) the dynamically Wiresized SERT has a delay of 6.95ns, a 31.6% improvement over statically-wiresized A-Tree. The SERT construction for this net has a "star" topology (i.e., direct wires from the source to all of the sinks).

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, *A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing*, Tech. Rep. CSD-TR-920051, Computer Science Department, UCLA, 1992.

[2] T. Barrera, J. Griffith, G. Robins, and T. Zhang, *Narrowing the Gap: Near-Optimal Steiner Trees in Polynomial Time*, in Proc. IEEE Intl. ASIC Conf., Rochester, NY, September 1993, pp. 87–90.

[3] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, *Fidelity and Near-Optimality of Elmore-Based Routing Constructions*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 1993, pp. 81–84.

[4] ——, *Towards Optimal Routing Trees*, in Proc. ACM/SIGDA Physical Design Workshop, Lake Arrowhead, CA, April 1993, pp. 44–51.

[5] K. D. Boese, A. B. Kahng, and G. Robins, *High-Performance Routing Trees With Identified Critical Sinks*, in Proc. ACM/IEEE Design Automation Conf., Dallas, June 1993, pp. 182–187.

[6] J. Cohoon and J. Randall, *Critical Net Routing*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 1991, pp. 174–177.

[7] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, *Provably Good Performance-Driven Global Routing*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 739–752.

[8] J. Cong, K. S. Leung, and D. Zhou, *Performance-Driven Interconnect Design Based on Distributed RC Delay Model*, in Proc. ACM/IEEE Design Automation Conf., Dallas, June 1993, pp. 606–611.

[9] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy, and R. I. McMillan, *Timing Driven Placement Using Complete Path Delays*, in Proc. ACM/IEEE Design Automation Conf., 1990, pp. 84–89.

[10] A. E. Dunlop, V. D. Agrawal, D. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, *Chip Layout Optimization Using Critical Path Weighting*, in Proc. ACM/IEEE Design Automation Conf., 1984, pp. 133–136.

[11] W. C. Elmore, *The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers*, J. Appl. Phys., 19 (1948), pp. 55–63.

[12] M. Garey and D. S. Johnson, *The Rectilinear Steiner Problem is NP-Complete*, SIAM J. Applied Math., 32 (1977), pp. 826–834.

[13] E. N. Gilbert and H. O. Pollak, *Steiner Minimal Trees*, SIAM J. Applied Math., 16 (1968), pp. 1–29.

[14] M. Hanan, *On Steiner's Problem With Rectilinear Distance*, SIAM J. Applied Math., 14 (1966), pp. 255–265.

[15] N. Hasan, G. Vijayan, and C. K. Wong, *A Neighborhood Improvement Algorithm for Rectilinear Steiner Trees*, in Proc. IEEE Intl. Symp. Circuits and Systems, New Orleans, LA, 1990.

[16] J.-M. Ho, G. Vijayan, and C. K. Wong, *New Algorithms for the Rectilinear Steiner Tree Problem*, IEEE Trans. Computer-Aided Design, 9 (1990), pp. 185–193.

[17] T. D. Hodes, B. A. McCoy, and G. Robins, *Dynamically-Wiresized Elmore-Based Routing Constructions*, in Proc. IEEE Intl. Symp. Circuits and Systems (to appear), London, England, May 1994.

[18] F. K. Hwang, *On Steiner Minimal Trees with Rectilinear Distance*, SIAM J. Applied Math., 30 (1976), pp. 104–114.

[19] ——, *An O(n log n) Algorithm for Rectilinear Minimal Spanning Trees*, J. ACM, 26 (1979), pp. 177–182.

[20] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, 1992.

[21] M. A. B. Jackson, E. S. Kuh, and M. Marek-Sadowska, *Timing-Driven Routing for Building Block Layout*, in Proc. IEEE Intl. Symp. Circuits and Systems, 1987, pp. 518–519.

[22] A. B. Kahng and G. Robins, *A New Family of Steiner Tree Heuristics With Good Performance: The Iterated 1-Steiner Approach*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 428–431.

[23] ——, *A New Class of Iterative Steiner Tree Heuristics With Good Performance*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 893–902.

[24] ——, *On Performance Bounds for a Class of Rectilinear Steiner Tree Heuristics in Arbitrary Dimension*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 1462–1465.

[25] S. Kim, R. M. Owens, and M. J. Irwin, *Experiments with a Performance Driven Module Generator*, in Proc. ACM/IEEE Design Automation Conf., June 1992, pp. 687–690.

[26] J. H. Lee, N. K. Bose, and F. K. Hwang, *Use of Steiner's Problem in Sub-Optimal Routing in Rectilinear Metric*, IEEE Trans. Circuits and Systems, 23 (1976), pp. 470–476.

[27] K. W. Lee and C. Sechen, *A New Global Router for Row-Based Layout*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 180–183.

[28] L. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, May 1975.

[29] S. Prasitjutrakul and W. J. Kubitz, *A Timing-Driven Global Router for Custom Chip Design*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 48–51.

[30] A. Prim, *Shortest Connecting Networks and Some Generalizations*, Bell Syst. Tech J., 36 (1957), pp. 1389–1401.

[31] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, *The Rectilinear Steiner Arborescence Problem*, Algorithmica, (1992), pp. 277–288.

[32] D. Richards, *Fast Heuristic Algorithms for Rectilinear Steiner Trees*, Algorithmica, 4 (1989), pp. 191–207.

[33] G. Robins, *On Optimal Interconnections*, Ph.D. Dissertation, CSD-TR-920024, Department of Computer Science, UCLA, 1992.

[34] J. Rubinstein, P. Penfield, and M. A. Horowitz, *Signal Delay in RC Tree Networks*, IEEE Trans. Computer-Aided Design, 2 (1983), pp. 202–211.

[35] T. L. Snyder, *On the Exact Location of Steiner Points in General Dimension*, SIAM J. Comput., 21 (1992), pp. 163–180.

[36] S. Sutanthavibul and E. Shragowitz, *An Adaptive Timing-Driven Layout for High Speed VLSI*, in Proc. ACM/IEEE Design Automation Conf., 1990, pp. 90–95.

[37] P. Winter, *Steiner Problem in Networks: A Survey*, Networks, 17 (1987), pp. 129–167.

[38] D. Zhou, S. Su, F. Tsui, D. S. Gao, and J. Cong, *Analysis of Trees of Transmission Lines*, Tech. Rep. CSD-TR-920010, Computer Science Department, UCLA, 1992.