Toward Optimal Routing Trees*

Kenneth D. Boese, Andrew B. Kahng, Bernard A. McCoy† and Gabriel Robins†

Computer Science Department, UCLA, Los Angeles, CA 90024-1596
† Computer Science Department, University of Virginia, Charlottesville, VA 22903-2442

Abstract

*We address the efficient construction of interconnection trees with near-*optimal *delay properties. We begin from first principles, and study the accuracy* **and fidelity** *of easily-computed delay models (specifically, Elmore delay) with respect to detailed simulation of underlying physical phenomena (e.g., SPICE-computed delays). Our studies show that minimization of Elmore delay is a high-fidelity interconnect objective within a range of IC interconnect technologies. We then propose a greedy* low delay tree *(LDT) heuristic which for any (monotone) delay function can efficiently minimize maximum delay. For comparison, we also generate* optimal routing trees *(ORTs) with respect to Elmore delay, using exhaustive search with branch-and-bound pruning. Experimental results show that the LDT heuristic approximates ORTs very accurately: for nets with up to seven pins, LDT trees have on average a maximum sink delay within 2% of optimum. Moreover, compared with traditional minimum spanning tree constructions, the LDT achieves* average *reductions in delay of up to 35% depending on the net size and technology parameters.*

1 Introduction

Over the last several decades, advances in VLSI fabrication technology have steadily improved the packing density of integrated circuits. As feature sizes decrease, device switching speeds tend to increase; however, smaller wire geometries imply higher resistance, so that signal propagation delay through the interconnect increases [17]. Thus, interconnection delay has had an increasing impact on circuit speed, and indeed it has been reported that interconnection delay contributes up to 70% of the clock cycle in the design of dense, high-performance circuits [20]. In light of this trend, performance-driven physical layout has become central to the design of leading-edge digital systems. Early work focused on performance-driven placement, with the usual objective being the close placement of cells in timing-critical paths, e.g., [8] [13] [14].

*Partial support for this work was provided by a GTE Graduate Fellowship, ARO DAAK-70-92-K-0001, ARO DAAL-03-92-G-0050, NSF MIP-9110696, and NSF MIP-9257982.

While timing-driven placement has a large effect on layout performance, the lack of optimal-delay interconnection algorithms will prevent designers from fully exploiting a high-quality placement. Certainly, once a module placement has been fixed, good timing-driven interconnection algorithms are key to enhancing the performance of the layout solution. For a given signal net, the typical objective has been to minimize the maximum signal delay to any sink. Many approaches have appeared in the literature, e.g., Dunlop et al. [9] determine net priorities based on static timing analysis, and process higher priority nets earlier using fewer feedthroughs; Jackson, Kuh and Marek-Sadowska [12] outline a hierarchical approach to timing-driven routing; and Prastjutrakul and Kubitz [16] use A* heuristic search and the Elmore delay formula [10] in their tree construction. Cong et al. have proposed finding minimum spanning trees with bounded source-sink pathlength [6], i.e., by simultaneously minimizing both tree cost and the maximum source-sink pathlength (i.e., tree radius); another cost-radius tradeoff was achieved by Alpert and coauthors [1]. More recently, Cong, Leung and Zhou [7] have shown that the use of rectilinear arborescence structures for interconnect topology design leads to substantial reduction in interconnection delays. Furthermore, Boese et al. [4] have developed a "critical sink" routing approach which significantly reduces delay to specified sinks, thereby exploiting the critical-path information available during iterative timing-driven layout.

The objective of our research is to identify and exploit a high-quality, algorithmically tractable model of interconnect delay. Previous methods have often relied on simple *abstractions*, e.g., geometric notions of "minimum tree cost", "bounded tree radius", or "low pathlength skew". Such models can simplify algorithm design, but may diverge from physical reality. We begin our work from "first principles": we exhaustively enumerate all routing solutions for particular signal nets using a range of interconnect technology parameters. Our goal is to determine a delay approximation that has both high accuracy *and high fidelity* with respect to physical models (i.e., SPICE-simulated delays).

In particular, we study the Elmore delay formula [10] and find it to be a high-fidelity routing objective: the minimum Elmore delay routing solution is very close in quality to the solution which minimizes SPICE-computed delay. Because exhaustive enumeration of all possible routing topologies is infeasible, we complement our studies of fidelity with a practical, *greedy* construction (the Low-Delay Tree, or LDT heuristic). According to our simulation results, the Elmore-based LDT solutions closely match (to within 2%, on average) the delays of Elmore-optimal solutions. LDT routings improve delays over those of traditional minimum spanning tree topologies by an average of up to 35%, depending on the size of the net and the technology parameters used.

2 Tree Delay Minimization

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 tree* $T(N)$, which is a spanning tree over N. Pin n_0 is the *source*, and the remaining pins are *sinks*. Each edge e_{ij} in $T(n)$ has an associated *edge cost*, d_{ij}, equal to the Manhattan distance between its two endpoints n_i and n_j; the *cost* of $T(n)$ 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 tree which minimizes the maximum source-sink delay:

Optimal Routing Tree (ORT) Problem: Given a signal net $N = \{n_0, n_1, ..., n_k\}$ with source n_0, construct a routing tree $T(N)$ such that $t(T(N)) = \max_{i=1}^{k} t(n_i)$ is minimized.[1]

[1] The ORT problem minimizes delay for individual nets without regard to the interdependence of nets in the overall circuit. In other words, the ORT problem concentrates on net-dependent objectives, rather than path-dependent objectives based on pre-defined critical paths. A path-dependent variant of the ORT problem can be defined by associating a *criticality* $\alpha_i \geq 0$ with each sink n_i reflecting timing information obtained during the performance-driven placement phase. The goal is then to construct a routing tree $T(N)$ which minimizes the weighted sum of the sink delays:

Critical-Sink Routing Tree (CSRT) Problem: Given a signal net $N = \{n_0, n_1, ..., n_k\}$ with source n_0 and possibly varying sink criticalities $\alpha_i \geq 0$, $i = 1, \ldots, k$, construct a routing tree $T(N)$ such that $\sum_{i=1}^{k} \alpha_i \cdot t(n_i)$ is minimized.

The CSRT problem formulation is quite general and captures traditional performance criteria for routing trees: (i) we can minimize average delay to all sinks by using all $\alpha_i \equiv$ some positive constant, then taking the L_1 sum of the weighted delays; and (ii) we can minimize the maximum delay to any sink by using all $\alpha_i \equiv$ some positive constant, then taking the L_∞ sum of the weighted delays. Yet a third variation can be used to solve the simple, yet realistic case where exactly one critical sink n_{CS} has been identified, i.e., $\alpha_{CS} = 1$ and all other $\alpha_i = 0$. The CSRT problem is studied in [4].

The specific routing tree that solves the ORT problem will depend on the method used to estimate delay. Ideally, we would like to compute and optimize delay according to the complete physical attributes of the circuit. To this end, the circuit simulator SPICE is generally regarded as the best available tool for obtaining precise estimates of interconnect delay. However, the computation times required by SPICE are too large for use during routing tree construction. The linear delay approximation has been used in the past [6] [20], but is known to be inaccurate. Thus, the Elmore delay formula [10] and the "Two-Pole" approximation developed by Zhou et al. [21] are both of interest, because they are more accurate than linear delay while also requiring less computation time than SPICE.

Elmore delay is defined as follows. Given routing tree $T(N)$ rooted at n_0, let e_i denote the edge from pin 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. Then the Elmore delay $t_{ED}(n_i)$ from source n_0 to sink n_i is computed as follows:

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

We can extend the t_{ED} function to entire trees by defining $t_{ED}(T(N)) = \max_{i=1}^{k} t_{ED}(n_i)$. If r_{e_j} and c_{e_j} are proportional to the length of e_j, the delay $t_{ED}(n_i)$ is quadratic in the length of the n_0-n_i path and also linear in total wirelength (which is proportional to C_{n_0}). Because of its relatively simple form, Elmore delay can be calculated in $O(k)$ time, as noted by Rubinstein et al. [19].

We note that the relative magnitude of the driver resistance r_d (i.e., versus unit wire resistance) can have a significant effect on the topology of the optimal routing tree: if r_d is large, the optimal routing tree is a minimum cost spanning tree, while if r_d is close to 0, the ORT will possess a "star" topology. Typical relative magnitudes of r_d are large for current generation CMOS, but decrease in, for example, submicron CMOS IC and MCM substrate interconnects.

Although Elmore delay has a compact definition and can be quickly computed, it does not capture all of the factors that account for delay. For example, the Two-Pole simulator of Zhou et al. [21] considers the impedance in a routing tree in addition to the capacitance and resistance modeled by the Elmore formula. According to [3] and [21], the Two-Pole simulator is intermediate between SPICE

and Elmore delay in both accuracy and speed of computation.

3 Accuracy and Fidelity of Delay Estimators

3.1 Accuracy

In choosing a delay simulator, one traditionally measures the *accuracy* of the available choices. The accuracy of a delay model is likely to vary with the circuit technology and the specifics of a net (for instance, the number of pins it contains, the size of the layout, etc.). Our first studies measure how close linear, Elmore, and Two-Pole delay estimates are to actual delay in a net.[2] We use nets of 4 to 7 pins using three technology files, representing three different resistance ratios. Table 1 gives parameters for three interconnect technologies which we call IC1, IC2, and IC3. (IC2 is representative of a typical 0.8μ CMOS process).

parameter	IC1	IC2	IC3
driver resistance (Ω)	10	100	1000
wire resistance ($\Omega/\mu m$)	0.03	0.03	0.03
wire capacitance ($fF/\mu m$)	0.352	0.352	0.352
wire inductance ($fH/\mu m$)	492	492	492
sink loading capacitance (fF)	15.3	15.3	15.3
layout area (mm^2)	10^2	10^2	10^2

Table 1: Parameter values for the three IC interconnect technologies.

Table 2 contains experimental results on the accuracy of the Elmore and Two-Pole models for each of the three IC technologies. The table shows the average ratio between SPICE delay and each of the two estimators; it also contains measures of the consistency of this ratio, in terms of both its standard deviation and 95%-confidence interval.[3] For each net size, the results are computed from 100 random nets connected using the minimum cost spanning tree (MST) construction. We use MSTs rather than random tree topologies so that our comparisons will be for relatively good (although not necessarily optimal) routing solutions; note that for these test sets, finding optimal-delay topologies using SPICE would be prohibitively time-consuming.

		$\|N\| = 4$		
		average	standard deviation	95% confidence
IC1	SPICE/Elmore	1.27	0.15	$\pm$ 0.32
	SPICE/2-Pole	0.48	0.05	$\pm$ 0.09
IC2	SPICE/Elmore	1.51	0.19	$\pm$ 0.37
	SPICE/2-Pole	0.64	0.09	$\pm$ 0.21
IC3	SPICE/Elmore	4.40	0.56	$\pm$ 2.06
	SPICE/2-Pole	2.22	0.31	$\pm$ 1.11

		$\|N\| = 7$		
		average	standard deviation	95% confidence
IC1	SPICE/Elmore	1.09	0.10	$\pm$ 0.20
	SPICE/2-Pole	0.47	0.03	$\pm$ 0.06
IC2	SPICE/Elmore	1.31	0.13	$\pm$ 0.26
	SPICE/2-Pole	0.60	0.06	$\pm$ 0.12
IC3	SPICE/Elmore	3.24	0.48	$\pm$ 0.92
	SPICE/2-Pole	1.67	0.26	$\pm$ 0.51

Table 2: Accuracy of the Elmore and Two-Pole estimators. The average ratio between "actual" SPICE delay and estimated delay is computed over 100 random nets with pin locations uniformly distributed over the layout area. The nets are connected using MST constructions. For each net size, we also compute the standard deviation and the 95% confidence interval of the ratios.

The results of Table 2 indicate that neither the Elmore or Two-Pole delay models give accurate estimates of delay. Only for 7-pin nets in IC1 is Elmore delay within 10% of SPICE on average; Two-Pole estimates of delay are not within 35% of SPICE on average for any of the net sizes and technologies tested. However, it should be noted that for each net size and technology, the Elmore and Two-Pole delay estimators are very consistent: the standard deviations and 95% confidence intervals for the are generally quite small, e.g., the standard deviations range from 8% to 16% of the average. Thus, use of precomputed "correction factors" may possibly compensate for the inaccuracy of these estimates.

3.2 Fidelity

The key observation underlying our work is that precise accuracy is not required of our delay estimates when using them to build routing trees. Rather, we require good estimators according to

[2] Again, we equate SPICE results with "actual delay". Our SPICE delay model uses constant resistance and capacitance values per unit of interconnect (i.e., both resistance and capacitance are proportional to wirelength). The root of the tree is driven by a resistor connected to the source. Thus we remove some physical characteristics of the driver in order to measure delay within the interconnect only, ignoring delay within the driver, which is dependent on the specific driver technology. Typically, a routing tree drives other CMOS devices; to model this, we attach uniformly-sized 2-transistor CMOS inverters to each pin. This is more realistic than using, e.g., pure capacitive pin loads, since the SPICE inverter model also captures the transient behavior associated with CMOS devices, which impacts signal propagation delay [15].

[3] The 95%-confidence interval is the smallest value $d > 0$ such that 95% of the sample ratios are within distance d of the average.

some measure of *fidelity*: i.e., how likely it is for an optimal or near-optimal routing solution according to a given estimator to also be nearly optimal according to actual (SPICE-simulated) delay. We define a measure of fidelity vis-a-vis an exhaustive enumeration of all possible routing solutions: we first rank all tree topologies by the given delay model, then rank the topologies again by SPICE delay, and then find the average difference between the two rankings for each topology. We have run simulations to estimate this measure of fidelity for nets of size 4 and 5 using the various delay estimators and each of the three IC technologies. (An early theorem of Cayley [11] implies that there are $|N|^{|N|-2}$ distinct spanning tree topologies for any given net N; see Figure 1 for the case $|N| = 4$.)

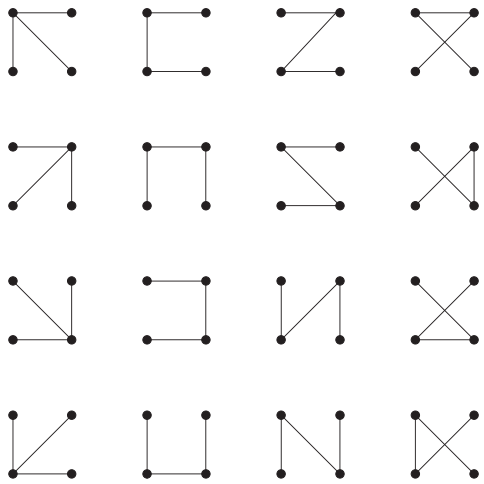

Figure 1: An inventory of all $4^{4-2} = 16$ tree topologies over 4 pins.

Table 3 assesses the fidelity to SPICE of the linear, Elmore, and Two-Pole delay estimators. We report the average difference in ranking over all topologies; the average rank difference for the topology which has lowest delay according to the estimator; and the average difference for the five topologies which have lowest delay according to the estimator. Our results show that Elmore delay has high fidelity, particularly when we compare the SPICE ranking of the optimal topology for Elmore delay with the optimal topology for linear delay: for nets of size 5 using technology IC3, optimal topologies under Elmore delay were on average 2.3 rank positions away from optimal according to SPICE. In comparison, the best topology under linear delay averaged distance 24.7 from its correct SPICE ranking. For 5-pin nets under the IC1 and IC2 technologies, the best topology under Elmore delay also has a near-optimal SPICE ranking: on average the distance from its SPICE ranking is 3.5 for IC1 (versus 4.6 under linear delay) and 1.5 for IC2 (versus 6.3 under linear delay).

		Linear vs SPICE		Elmore vs SPICE	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	All	0.84	6.44	0.71	3.30
	Best	0.70	4.61	0.65	3.50
	5 Best	1.00	3.58	1.11	3.54
IC2	All	1.33	8.69	0.82	4.75
	Best	2.05	6.25	0.70	1.45
	5 Best	1.47	6.24	1.05	3.30
IC3	All	2.57	23.01	0.43	9.18
	Best	3.10	24.65	0.05	2.30
	5 Best	2.84	29.60	0.39	3.08

		2-Pole vs SPICE		Elmore vs 2-Pole	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	All	0.48	1.71	0.66	3.37
	Best	0.40	1.50	0.45	2.50
	5 Best	0.75	1.86	1.02	3.63
IC2	All	0.44	2.93	0.56	2.83
	Best	0.10	0.45	0.55	0.90
	5 Best	0.53	1.24	0.86	2.49
IC3	All	0.44	9.46	0.23	1.57
	Best	0.05	2.30	0.00	0.00
	5 Best	0.38	3.17	0.13	0.35

Table 3: Average difference in rankings of topologies according to different delay models. The sample consists of 20 random nets of each cardinality. Note that the total number of topologies for each net is 16 for $|N| = 4$ and 125 for $|N| = 5$.

For IC1, the difference of 3.5 positions leads to an average 12.4% penalty in SPICE-computed delay. This can be seen from Table 4, which shows the drop-off in IC1 SPICE delay quality for each rank, when compared with optimal delay. For IC2 (an actual 0.8μ CMOS process), the distance of 1.5 positions implies a difference of approximately 6.6% in actual SPICE-computed delay.

Table 3 shows that the Two-Pole simulator has somewhat better fidelity than Elmore delay. However, the relatively small improvement in fidelity does not seem to justify the much greater computation that is required to search over solution topologies using Two-Pole as opposed to using the linear-time Elmore delay computation.

4 Near-Optimal Routing Trees

We can solve the ORT problem *optimally* for any delay model using a backtracking enumeration of tree topologies with branch-and-bound pruning. Starting with a trivial tree containing only the source pin, we incrementally add one edge at a time to the growing tree. At each step we compute the maximum delay from the source to any sink in the tree. If this value exceeds the maximum delay of any *complete* candidate tree seen so far, we

IC1				
1-25	26-50	51-75	76-100	101-125
1.000	1.651	2.341	3.150	4.102
1.042	1.671	2.382	3.190	4.127
1.083	1.679	2.392	3.223	4.177
1.114	1.725	2.415	3.234	4.221
1.136	1.771	2.450	3.262	4.275
1.150	1.782	2.476	3.307	4.357
1.181	1.794	2.501	3.352	4.413
1.219	1.805	2.570	3.365	4.479
1.238	1.841	2.590	3.391	4.567
1.261	1.856	2.636	3.416	4.633
1.272	1.876	2.662	3.441	4.717
1.283	1.895	2.702	3.494	4.754
1.313	1.942	2.713	3.511	4.817
1.329	1.998	2.758	3.544	4.905
1.337	2.019	2.799	3.591	4.990
1.364	2.051	2.815	3.621	5.064
1.415	2.091	2.844	3.677	5.183
1.452	2.112	2.913	3.706	5.248
1.478	2.140	2.931	3.759	5.325
1.495	2.169	2.947	3.800	5.422
1.508	2.194	2.982	3.845	5.530
1.537	2.227	3.015	3.898	5.788
1.574	2.268	3.061	3.956	6.027
1.596	2.294	3.101	3.994	6.380
1.619	2.326	3.121	4.045	6.665

Table 4: SPICE delay ratios of all 125 topologies for $|N| = 5$ using IC1 technology parameters. All values are normalized to the delay value for the best topology, and are averaged over 20 random sets of pin locations.

Branch-and-Bound
Optimal Routing Tree (BBORT) Method

Input: signal net N with source $n_0 \in N$
Output: optimal-delay tree T_{opt} over N

1. $T = (V, E) = (\{n_0\}, \emptyset)$
2. $t_{\min} = \infty$
3. **Call** Add_Edges(T)
4. **Output** T_{opt}

Procedure Add_Edges(Tree: $T = (V, E)$)
5. **While** there exist $v \in V$ and $u \notin V$ such that $T' = (V \cup \{u\}, E \cup \{(u, v)\})$ is a new tree topology **Do**
6. **Compute** tree delay $t(T')$
7. **If** $t(T') \leq t_{\min}$ **Then**
8. **If** $|T'| = |N|$ **Then** $T_{opt} = T'$, $t_{\min} = t(T')$
9. **Else Call** Add_Edges(T')

Figure 2: The branch-and-bound ORT template (recursive implementation).

prune the search and backtrack to select a different edge at the previous step. Figure 2 depicts a recursive implementation of this *Branch-and-Bound ORT* (BBORT) search.

BBORT will find the optimal-delay tree as long as the delay function possesses a *monotonicity* property, i.e., the tree delay does not decrease with the addition of a new edge. The number of topologies considered can be further reduced by initializing the value of t_{min} in Figure 2 to the maximum source/sink delay of some "good" heuristic routing tree over N. Despite this pruning of the solution space, however, the worst-case time complexity of BBORT is still exponential.

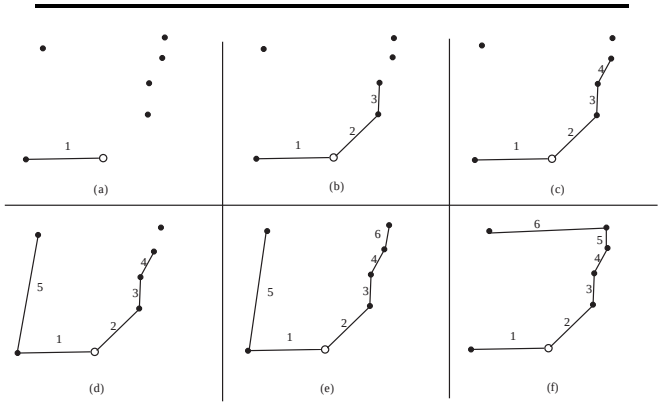

Figure 3: (a)-(e): A growing LDT. (f): An MST on the same net.

To avoid the exponential running time of exhaustive enumeration, we propose the following greedy heuristic to approximate ORTs. Our method is analogous to Prim's minimum spanning tree construction [18]: starting with a trivial tree containing only the source, we iteratively find a pin n_i in the tree and a sink n_j outside the tree so that adding edge e_{ij} yields a tree with minimum delay. The construction terminates when the entire net is spanned by the growing tree. Pseudo-code for this *Low Delay Tree* (LDT) algorithm is given in Figure 4.

Low Delay Tree (LDT) Heuristic

Input: signal net N with source $n_0 \in N$
Output: low-delay routing tree T over N

1. $T = (V, E) = (\{n_0\}, \emptyset)$
2. **While** $|V| < |N|$ **Do**
3. **Find** $n_i \in V$ and $n_j \notin V$ minimizing the tree delay $t((V \cup \{u\}, E \cup \{e_{ij}\}))$
4. $V = V \cup \{n_j\}$
5. $E = E \cup \{e_{ij}\}$
6. **Output** resulting spanning tree $T = (V, E)$

Figure 4: The Low Delay Tree heuristic: a greedy approximation of optimal routing trees.

The LDT heuristic may be viewed as generalizing the Elmore Routing Tree algorithm of Boese, Kahng and Robins [4] to any given delay model. If the de-

lay at all pins can be calculated in $O(k)$ time (as is the case with Elmore delay), then LDT can easily be implemented in $O(k^3)$ time by using the following observation[4]: if a new tree edge incident to sink $v \in V$ (Line 3 of Figure 4) minimizes the maximum delay $\max_i t_{ED}(n_i)$, in general it must connect v to the sink $u \notin V$ that is closest to v. Consequently, at each pass through the while loop in Figure 4, we can update the shortest "outside connections" for every $v \in V$ (in time $O(k^2)$ in the worst-case), and then simply add each of these $O(k)$ outside connections to T in turn. The delays to all sinks of the resulting trees can be evaluated in $O(k)$ time per tree. We then choose the outside connection that results in the least increase in tree delay. Hence, each pass through the while loop requires $O(k^2)$ time, yielding the $O(k^3)$ complexity result. In practice this time complexity is not a hindrance, since k is small. As shown in the next section, Elmore-based LDTs have delay within 2.3% of optimal Elmore-delay trees; in combination with our studies of fidelity, this provides strong evidence that the LDT heuristic produces trees of near-optimal quality.

5 Experimental Results

We have implemented both the BBORT and LDT methods, based on Elmore delay and using C in the UNIX/Sun environment. We have run trials on sets of 500 nets for each of several net sizes; pin locations were randomly chosen from a uniform distribution in a square layout region. Our inputs correspond to the same IC parameters studied in Section 3.

Table 5 compares Elmore delays of the Elmore-based ORT (i.e., BBORT) and LDT constructions, and of the minimum spanning tree (MST) and shortest path tree (SPT) constructions, for the IC1 technology.[5] Delay for each tree is normalized to the ORT delay of the same net. Wirelengths are similarly compared, with the cost of each tree normalized to the MST cost of the net. Tables 6 and 7 give analogous results for the IC2 and IC3 technology parameters.

In Table 5 we see that under the IC1 technology, LDTs over 7 pins have an average maximum Elmore delay only 1.1% greater than optimal, while MSTs have delay 124% greater than optimal on average. For smaller nets, LDTs are even closer to optimal: for nets with 4 pins, LDT delays are only 0.9% above optimal on average, while MSTs are 41.6% above optimal. Our confidence in the average difference computed between LDTs and ORTs is very high: for instance, the 1.1% difference obtained for

IC1 (delay)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000
LDT	1.009	1.059	1.008	1.025	1.011	1.037
SPT	1.009	1.059	1.028	1.199	1.094	1.540
MST	1.416	1.907	1.708	2.745	2.237	4.056

IC1 (cost)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.288	1.604	1.367	1.797	1.466	1.810
LDT	1.288	1.604	1.395	1.797	1.466	1.892
ORT	1.209	1.520	1.286	1.571	1.444	1.326

Table 5: Elmore delays and wirelengths of various constructions using using IC1 parameters. Simulations were run on 500 random nets for each net size. Cost values are normalized to MST cost and tree delays are normalized to the (Elmore-based) ORT delay. Standard errors for LDT-Elmore are 0.0006 for $|N| = 4$; 0.0003 for $|N| = 5$; and 0.0004 for $|N| = 7$.

7 pins has a standard error[6] of 0.04%, indicating a 95% confidence interval between 1.0% and 1.2% (i.e., an interval of within two times the standard error of the average). Even in the worst case, LDTs are close to optimal: over 500 random nets, the highest difference between LDT and ORT delays is only 5.9% for 4-pin nets and 3.7% for 7-pin nets. The high performance of LDTs is achieved with an average wirelength penalty compared to MSTs that ranges from 28.8% for 4-pin nets to 46.6% for 7-pin nets.

Table 6 contains what seem to be our worst results in terms of the optimality of LDTs. For the IC2 parameters and 7-pin nets, LDT gives an average value within 2.3% of ORT with a 95% confidence interval of 2.0% to 2.6%. In Table 7, we see that the Elmore-based LDT constructions are very close to optimal for IC3 parameters: they are on average within 0.5% of ORT delay for 7-pin nets. Note that for IC3, the MST performance improves significantly, while the SPT performance worsens. By contrast, the LDT algorithm produces very good results for each of the three technologies, as is expected since it optimizes Elmore delay directly.

Table 8 compares delays in Elmore-based LDTs with those of the MST and AHHK [1] constructions for nets with up to 17 pins under the IC2 technology. The AHHK algorithm of Alpert et al. is a re-

[4] Note that this observation assumes constant loading capacitances, unit resistances, and unit capacitances for the Elmore model.

[5] The SPT construction is the tree which minimizes cost subject to each source/sink path having minimum length.

[6] As used here, the term *standard error* is defined as follows. For a random variable X, let $\hat{X} = \sum_{i=1}^{n} X_i$ be an estimator for the expected value of X. The standard error of $\hat{X}$ is an estimate of its standard deviation over multiple sample sets, and is equal to the standard deviation of X divided by $\sqrt{n}$. Because delays are recorded as ratios to the ORT delay, the standard error of the average difference between LDT and ORT delays is equivalent to the standard error of average LDT delay.

IC2 (delay)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000
LDT	1.003	1.114	1.010	1.147	1.023	1.164
SPT	1.033	1.280	1.061	1.365	1.114	1.555
MST	1.165	2.370	1.240	2.375	1.381	2.960

IC2 (cost)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.207	2.106	1.283	2.605	1.381	2.725
LDT	1.103	1.666	1.147	1.917	1.201	1.731
ORT	1.100	1.666	1.131	1.652	1.162	1.673

Table 6: Simulation results using IC2 parameters. Standard errors for the average delay difference between LDT and ORT are 0.0006 for $|N| = 4$; 0.0010 for $|N| = 5$; and 0.0014 for $|N| = 7$.

IC3 (delay)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000
LDT	1.0003	1.035	1.001	1.051	1.005	1.061
SPT	1.142	1.844	1.120	2.226	1.268	2.377
MST	1.007	1.161	1.014	1.170	1.025	1.208

IC3 (cost)	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.207	2.106	1.283	2.605	1.381	2.273
LDT	1.006	1.139	1.010	1.142	1.012	1.143
ORT	1.006	1.139	1.012	1.140	1.019	1.722

Table 7: Simulation results using IC3 parameters. Standard errors for the average delay difference between LDT and ORT are 0.0001 for $|N| = 4$; 0.0003 for $|N| = 5$; and 0.0005 for $|N| = 7$.

cent cost-radius tradeoff construction which yields less tree cost (and signal delay) for given tree radius bounds when compared with the BRBC construction of Cong et al.[6]. Each value in a given column represents an average over the same set of 500 random signal nets. Data shown include average tree delay, maximum tree delay, the respective delay ratios, and average tree costs. Because of the size of this test set, all delays in Table 8 are calculated using the Two-Pole simulator. Our results indicate that the LDT algorithm is highly effective for larger nets, and also outperforms the best known direct tradeoff between tree radius and cost (i.e., AHHK). For nets with 16 sinks, the LDT construction reduces average sink delay by 35% compare to MSTs and by 6.2% compared to AHHK trees.

IC2				
		$\|N\| = 5$	$\|N\| = 9$	$\|N\| = 17$
Ave. Tree Delay (ns)	MST	3.72	5.58	8.37
	SPT	3.28	4.49	6.31
	AHHK	3.24	4.31	5.77
	LDT	3.11	4.11	5.41
Tree Delay Ratios	LDT/MST	.836	.737	.646
	LDT/AHHK	.960	.954	.938
Average wirelength (cm)	MST	1.65	2.43	3.46
	SPT	2.14	3.51	5.53
	AHHK	1.84	2.75	4.05
	LDT	1.91	2.99	4.32

Table 8: Simulation results for IC2 comparing LDT with MST and the AHHK algorithm on nets with up to 17 pins. Averages in each column are taken over 500 signal nets with pin locations chosen randomly from the layout region. Reported delays are all calculated using the Two-Pole simulator.

6 Conclusions and Future Directions

Many previous approaches to interconnect delay minimization have been hampered by their ad hoc selection and use of delay estimates in the routing construction. To find an easily computed delay estimate for use in constructing a high-quality interconnection tree, we begin from first principles. We have addressed the issue of the accuracy and *fidelity* of the Elmore [10] and Two-Pole [21] delay models by comparing the rankings of tree topologies according to these estimates with rankings according to the SPICE simulator. Our studies indicate that algorithms which minimize the Elmore and Two-Pole delay estimates should also effectively minimize actual delay. We have also used the branch-and-bound BBORT method to determine *optimal* routing trees for any given monotonic delay function.

To achieve a practical and near-optimal routing methodology, we have proposed the greedy *Low Delay Tree* (LDT) heuristic. LDT can be implemented using any given model of delay; because of the demonstrated fidelity of Elmore delay, we have implemented LDT using that model. Experimental results show that LDT performs essentially as well as exhaustive search on nets with up to 7 pins. In addition, for large sets of benchmarks, LDT achieves reductions in delay of up to 35% (depending on circuit technology and net size) over the MST routing, as measured by the Two-Pole simulator.

Our LDT algorithm is formulated to construct a spanning tree, but can easily be extended to yield a *Steiner Low Delay Tree* (SLDT) algorithm. For example, we may allow each newly selected pin to connect to an arbitrary point in an existing tree edge, possibly inducing a Steiner point. Simulation results in [4] indicate that the SLDT algorithm using Elmore delay is also highly effective. LDT can also be generalized to "critical-sink routing" (recall Footnote 1) by modifying the objective function in the LDT and SLDT algorithms to minimize delay at prescribed critical sinks [4]. Furthermore, our constructions can be adapted to minimize maximum

tree delay, average tree delay (i.e., sum of delays to all the pins), or any other well-behaved delay function.

Because the typical CAD environment consists of a large network of workstations and servers, there is tremendous potential for improvement of running times through parallel/distributed implementations [2] [5]. We note that algorithms described in this paper are highly parallelizable, e.g. the BBORT method can use p processors to simultaneously explore routing topologies in different regions of the solution space. Similarly, the LDT algorithm can employ separate processors to determine the effects on delay of adding different candidate edges to the growing routing topology.

7 Acknowledgements

We are grateful to the authors of [21] for use of their simulator code. Many thanks go to Professors Andy Schwab, Hugh Landes, and Michael Shur of the University of Virginia Electrical Engineering Department for their help with SPICE.

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*, in to appear in Proc. IEEE Intl. Symp. on Circuits and Systems, Chicago, IL, May 1993.

[2] T. Barrera, J. Griffith, S. A. McKee, G. Robins, and T. Zhang, *Toward a Steiner Engine: Enhanced Serial and Parallel Implementations of the Iterated 1-Steiner Algorithm*, in Great Lakes Symposium on VLSI, Kalamazoo, MI, March 1993, pp. 90–94.

[3] K. D. Boese, J. Cong, A. B. Kahng, K. S. Leung, and D. Zhou, *On High-Speed VLSI Interconnects: Analysis and Design*, Proc. Asia-Pacific Conf. on Circuits and Systems, (1992), pp. 35–40.

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

[5] R. J. Brouwer and P. Banerjee, *PHIGURE: A Parallel Hierarchical Global Router*, in Proc. ACM/IEEE Design Automation Conf., 1990, pp. 650–653.

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

[7] J. Cong, K. S. Leung, and D. Zhou, *Performance-Driven Interconnect Design Based on Distributed RC Delay Model*, in to appear in ACM/IEEE Design Automation Conf., Dallas, June 1993.

[8] 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.

[9] 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.

[10] 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.

[11] S. Even, *Graph Algorithms*, Computer Science Press, Inc., Potomac, MD, 1979.

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

[13] I. Lin and D. H. C. Du, *Performance-Driven Constructive Placement*, in Proc. ACM/IEEE Design Automation Conf., 1990, pp. 103–106.

[14] M. Marek-Sadowska and S. P. Lin, *Timing Driven Placement*, in Proc. IEEE Intl. Conf. on Computer-Aided Design, Santa Clara, CA, November 1989, pp. 94–97.

[15] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.

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

[17] B. T. Preas and M. J. Lorenzetti, *Physical Design Automation of VLSI Systems*, Benjamin/Cummings, Menlo Park, CA, 1988.

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

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

[20] 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.

[21] D. Zhou, F. Tsui, J. Cong, and D. Gao, *A Distributive RCL-Model for MCM Layout*, in IEEE Multi-Chip Module Conf., March 1993, pp. 191–197.