Fidelity and Near-Optimality of Elmore-Based Routing Constructions

K. D. Boese,A. B. Kahng,B. A. McCoy,G. Robins

Technical Report No. CS-93-14
April 06, 1993

Fidelity and Near-Optimality of Elmore-Based Routing Constructions*

K. D. Boese, A. B. Kahng, B. A. McCoy† and G. Robins†

Computer Science Department, University of California at Los Angeles, 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. Our study begins from first principles: we consider the accuracy *and fidelity* of easily-computed delay models (specifically, Elmore delay) with respect to the delay values computed from detailed simulation of underlying physical phenomena (e.g., SPICE simulator output). Our studies show that minimization of Elmore delay is a high-accuracy, high-fidelity interconnect objective within a range of IC interconnect technologies. We propose an efficient *low delay tree* (LDT) heuristic which uses a greedy construction to minimize maximum sink delay for a given (monotone) delay function. For comparison purposes, we also generate *optimal routing trees* (ORTs) under 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, thinner wire has higher resistance, so that signal propagation delay through the interconnect increases [15]. According to the ideal CMOS scaling rules, decreasing device dimensions by a factor of $\frac{1}{\Delta}$ and increasing the chip area by a factor of α causes a decrease in gate delay by a factor of Δ, and an increase in global net delay by a factor of $\Delta^2 \cdot \alpha^2$ [19]. 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

*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..

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., [6] [11] [12].

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 from the source pin to any sink pin. Many approaches have appeared in the literature, e.g., Dunlop et al. [7] determine net priorities based on static timing analysis, and process higher priority nets earlier, using fewer feedthroughs; Jackson, Kuh and Marek-Sadowska [10] outline a hierarchical approach to timing-driven routing; and Prastjutrakul and Kubitz [14] use A* heuristic search and the Elmore delay formula [8] in their tree optimization. Cong et al. have proposed finding minimum spanning trees with bounded source-sink pathlength [5], i.e., by simultaneously minimizing both tree cost and the maximum source-sink pathlength (i.e., tree radius); a similar cost-radius tradeoff was achieved by [1]. More recently, Boese et al. [3] have developed a "critical sink" routing approach which significantly reduces delay to specified sinks, thereby exploiting the critical-path information that is implicitly 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 is of both high accuracy *and high fidelity* with respect to physical models (i.e., SPICE-simulated delays). Our studies show that the Elmore delay formula [8] is a high-fidelity routing objective: the minimum Elmore delay solution is very close in quality to the minimum SPICE-computed delay solution. Because exhaustive enumeration of the Elmore delays for 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-base LDT solutions closely approximate (to within 2%, on average) 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.

The rest of our paper is organized as follows. Section 2 gives basic definitions and formalizes the problem of constructing an optimal-delay interconnection tree. In Section 3 we present experimental results on the fidelity and accuracy of the Elmore delay model with respect to SPICE simulations. Section 4 describes a branch-and-bound method for constructing the optimal routing tree under a given delay model, and also presents the LDT heuristic for approximating optimal routing trees. In Section 5, we provide experimental results on the performance of and Elmore-based LDT implementation. Section 6 concludes with extensions to the LDT method and directions for future research.

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* $T(N)$. Pin n_0 is a *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]

The specific routing tree that solves the ORT problem will depend on the model 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, we might use the circuit simulator SPICE, which is generally regarded as the best available tool for obtaining a precise, complete measure of interconnect delay. However, if we wish to use delay estimates dynamically during the global routing phase, the computation time

[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 as follows. For each sink n_i in N we can associate a *criticality* α_i, reflecting the timing information obtained during the performance-driven placement phase. Our goal is to construct a routing tree $T(N)$ which minimizes the weighted sum of the sink delays:

The 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. For example, it 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 for one critical sink is studied in [3].

required by SPICE will be too large. The linear delay approximation has been used in the past [5] [20], but is known to be inaccurate. The Elmore delay formula [8] and the "Two-Pole" approximation developed by Zhou et al. [21] are two estimators that are theoretically more accurate than linear delay, yet easier to compute 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:[2]

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

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. We note that the driver resistance r_d 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 approaches 0, then the ORT will possess a "star" topology. The size of r_d relative to unit wire resistance is a "resistance ratio" that an interconnect technology vis-a-vis routing tree design. Typical values 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 [2] and [21], the Two-Pole simulator is intermediate between SPICE and Elmore delay in both accuracy and speed of computation.

[2]Because of its relatively simple form, Elmore delay can be calculated in $O(k)$ time, as noted by Rubinstein et al. [18]. The calculation can be accomplished using two depth-first traversals: 1) to compute the delay along each edge and 2) to sum up the delays along each source/sink path.

3 Fidelity of Three 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 (again, we equate SPICE results with "actual delay"). We use nets of 4 to 7 pins using three technology files, representing three different resistance ratios. Table 1 shows the specific parameters of three IC 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	10 Ω	100 Ω	1000 Ω
wire resistance	0.03 $\Omega/\mu m$	0.03 $\Omega/\mu m$	0.03 $\Omega/\mu m$
wire capacitance	0.352 $fF/\mu m$	0.352 $fF/\mu m$	0.352 $fF/\mu m$
wire inductance	492 $fH/\mu m$	492 $fH/\mu m$	492 $fH/\mu m$
sink loading capacitance	15.3 fF	15.3 fF	15.3 fF
layout area	10^2 mm^2	10^2 mm^2	10^2 mm^2

Table 1: Parameter values for the various CMOS interconnect technologies.

Accuracy of Elmore and Two-Pole Estimators for MST Constructions							
		$\lvert N\rvert = 4$			$\lvert N\rvert = 7$		
		average	standard deviation	95% confidence	average	standard deviation	95% confidence
IC1	SPICE/Elmore	1.27	0.15	$\pm$ 0.32	1.09	0.10	$\pm$ 0.20
	SPICE/2-Pole	0.48	0.05	$\pm$ 0.09	0.47	0.03	$\pm$ 0.06
IC2	SPICE/Elmore	1.51	0.19	$\pm$ 0.37	1.31	0.13	$\pm$ 0.26
	SPICE/2-Pole	0.64	0.09	$\pm$ 0.21	0.60	0.06	$\pm$ 0.12
IC3	SPICE/Elmore	4.40	0.56	$\pm$ 2.06	3.24	0.48	$\pm$ 0.92
	SPICE/2-Pole	2.22	0.31	$\pm$ 1.11	1.67	0.26	$\pm$ 0.51

Table 2: For each net, we compute the ratio between "actual" SPICE delay and the estimated delay. For each net size, we compute the average ratio, 95% confidence interval (i.e., smallest interval from average containing 95% of the ratios), and standard deviation of the ratio. over 100 random nets with pin locations uniformly distributed over the layout area.

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. In reality, a routing tree typically 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 [13].

Table 2 show the average ratio between SPICE and the Elmore and Two-Pole models for each IC technology. The table also contains measures of the consistency of this ratio, in terms of both its standard deviation and 95%-confidence interval. (The confidence interval is defined as the smallest distance from the average such that 95% of the samples are contained within this interval.) 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 topologies so that our comparisons will be for good (although not necessarily optimal) interconnections. (It would be prohibitively time-consuming to find optimal-delay topologies using SPICE.)

Even if two delay estimators return very different numbers, we say that each is accurate with respect to the other as long as the ratio of the two delay estimates is essentially constant. The results in Table 2 show that for the most of the IC technologies, both the Elmore and Two-Pole models are highly accurate in their estimates of SPICE delay. Note for most technologies and pin sizes, the SPICE/Elmore ratio has standard deviations ranging 8% - 16% and have good confidence intervals of between $\pm 18\%$ and $\pm 26\%$. The Two-Pole estimates are also highly accurate for most technologies. Additionally, the level of accuracy of both models is consistent over all three IC technologies.

3.2 Fidelity

The key observation underlying our work is that precise accuracy is not required of our delay estimates when we use them to build routing trees. Rather, we require good estimators according to 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 physical (SPICE-simulated) delay. We can define a measure of fidelity by first ranking all tree topologies by the given delay model, ranking again by SPICE delay, and then finding the average difference between the two rankings for each topology. (An early theorem of Cayley [9] implies that there are $|N|^{|N|-2}$ distinct spanning tree topologies for any given net N; see Figure 1 for the case $|N| = 4$.) We have run simulations to estimate this measure of fidelity for Elmore delay for nets of size 4 and 5 using each of the three IC technologies.

Table 3 assesses the fidelity to SPICE of the rankings of tree topologies according to the linear and Elmore delay models. We report the average difference in ranking over all topologies; the average difference for the topology which has lowest delay according to the first estimate; and the average

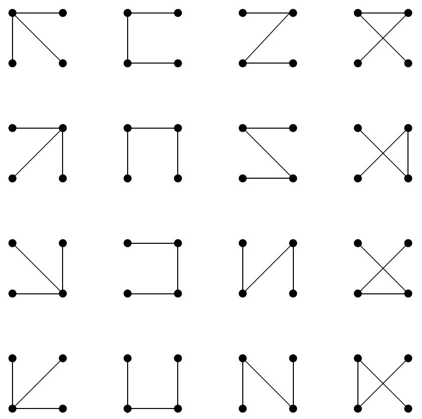

Figure 1: An inventory of all $4^{4-2} = 16$ tree topologies over 4 pins.

		Linear vs SPICE $|N| = 4$	Linear vs SPICE $|N| = 5$	Elmore vs SPICE $|N| = 4$	Elmore vs SPICE $|N| = 5$
IC1	All Topologies	3.81	6.60	3.40	3.61
	Best Topology	4.00	5.28	0.70	2.57
	5 Best Topologies	3.56	3.40	2.62	3.88
IC2	All Topologies	1.99	8.62	1.96	4.92
	Best Topology	3.55	11.33	1.56	1.87
	5 Best Topologies	2.54	9.14	1.91	3.78
IC3	All Topologies	2.54	22.79	0.43	8.71
	Best Topology	3.00	29.29	0.05	0.00
	5 Best Topologies	2.82	32.33	0.39	0.66

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 $|N| = 5$.

difference for the five topologies which have lowest delay according to the first estimate. 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 *always* optimal according to SPICE in our sample! In comparison, the best topology under linear delay was distance 29 away on average from its correct SPICE ranking. For 5-pin nets under the IC1 and IC2 technologies, best topology under Elmore delay also has a very high SPICE ranking: on average the distance from its SPICE ranking is 2.6 for IC1 (versus 5.3 under linear delay) and 1.9 for IC2 (versus 11.3 under linear delay). [3] We have

[3]For IC2, the distance of 1.87 positions implies a difference of approximately 3.8% in actual SPICE-computed delay. For IC1, the difference of 2.57 positions leads to an average 8.57% penalty in SPICE-computed delay. In the Appendix, we give three tables showing the drop-off in SPICE delay quality for each rank, when compared with optimal delay.

also found somewhat better fidelity results for the Two-Pole simulator. However, the relatively small improvement in fidelity may not justify the much greater amount of computation required to search over solution topologies using the Two-Pole simulator instead of the (linear-time) Elmore computation.

4 Near-Optimal Routing Trees

The ORT problem can be solved optimally for any delay model by 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 may 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.

```
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
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).

BBORT will find the optimal-delay tree as long as the delay function possesses a *monotonicity* property, i.e., the maximum 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. However, despite this pruning of the solution space, the worst-case time complexity of BBORT is exponential.

In order to avoid the exponential running time of exhaustive enumeration, we propose the following greedy heuristic to approximate ORTs. Our method is an analogue of Prim's minimum spanning tree

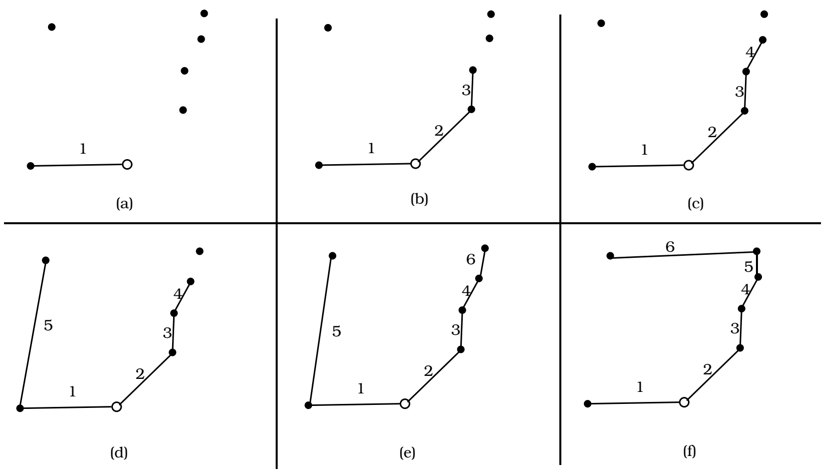

Figure 3: (a)-(e): A growing LDT. (f): An MST on the same net.

construction [16]: 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 the 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 [3] to any given delay model. If the delay 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.[4] In

[4]The $O(k^3)$ time bound is achieved by using the following observation: 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.

practice this time complexity is not a hindrance, since k is small. Our experimental results in Section 5 confirm that Elmore-based LDTs have delay within 2.3% of optimal Elmore-delay trees, providing strong evidence that the LDT heuristic produces trees of near-optimal quality.

5 Experimental Results: Greedy LDT is Near-Optimal

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 the square layout region. Our inputs correspond to the same range of IC parameters studied in Section 3.

Table 4 compares Elmore delays of the Elmore-based BBORT and LDT constructions, as well as the minimum spanning tree (MST) and shortest path tree (SPT)[5] constructions for the IC1 technology. Maximum delay for each tree is normalized to the ORT delay on the same net. Wirelengths are similarly compared, with the cost of each tree normalized to the MST cost of the net. Tables 5 through 6 give the analogous results for the IC2 and IC3 technology parameters.

IC1	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 6$		$\|N\| = 7$	
(delay)	ave	max	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
LDT-Elmore	1.009	1.059	1.008	1.025	1.011	1.024	1.011	1.037
SPT	1.009	1.059	1.028	1.199	1.058	1.307	1.094	1.540
MST	1.416	1.907	1.708	2.745	1.908	2.934	2.237	4.056

IC1	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 6$		$\|N\| = 7$	
(cost)	ave	max	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.288	1.604	1.367	1.797	1.475	1.990	1.466	1.810
LDT-Elmore	1.288	1.604	1.395	1.797	1.451	1.667	1.466	1.892
ORT	1.209	1.520	1.286	1.571	1.317	1.542	1.444	1.326

Table 4: Elmore delays and wirelengths of various constructions using using IC1 parameters. The simulations were run over 500 random nets for each net size. For each net, cost values are normalized to MST cost and tree delays are normalized to the (Elmore-based) ORT delay. Standard errors for the average delay difference between LDT-Elmore and ORT are 0.0006 for $|N| = 4$; 0.0003 for $|N| = 5$; 0.0003 for $|N| = 6$; and 0.0004 for $|N| = 7$.

In Table 4 we see that, under the IC1 technology, LDTs over 7 pins have an average maximum Elmore delay only 1.1% greater than optimal, while on average MSTs have delay 124% greater than

[5] The SPT construction is the tree which minimizes cost subject to each source/sink path having minimum length. Because we use the Manhattan geometry, SPTs will not always have a star topology.

optimal. For smaller nets, delays of 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 7 pins has a standard error 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 from the average). Even in the worst case, LDTs are close to optimal: over 500 random 4-pin nets, the highest difference between LDT and ORT delays is only 5.9%; for 7-pin nets, the maximum difference is even lower at 3.7%. 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.

IC2	$\lvert N\rvert = 4$		$\lvert N\rvert = 5$		$\lvert N\rvert = 6$		$\lvert N\rvert = 7$	
(delay)	ave	max	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
LDT	1.003	1.114	1.010	1.147	1.017	1.141	1.023	1.164
SPT	1.033	1.280	1.061	1.365	1.087	1.495	1.114	1.555
MST	1.165	2.370	1.240	2.375	1.312	2.360	1.381	2.960

IC2	$\lvert N\rvert = 4$		$\lvert N\rvert = 5$		$\lvert N\rvert = 6$		$\lvert N\rvert = 7$	
(cost)	ave	max	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.207	2.106	1.283	2.605	1.342	2.775	1.381	2.725
LDT	1.103	1.666	1.147	1.917	1.180	2.042	1.201	1.731
ORT	1.100	1.666	1.131	1.652	1.152	1.673	1.162	1.673

Table 5: 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 $|N| = 5$; 0.0013 for $|N| = 6$; and 0.0014 for $|N| = 7$.

IC3	$\lvert N\rvert = 4$		$\lvert N\rvert = 5$		$\lvert N\rvert = 6$		$\lvert N\rvert = 7$	
(delay)	ave	max	ave	max	ave	max	ave	max
ORT	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
LDT	1.0003	1.035	1.001	1.051	1.003	1.067	1.005	1.061
SPT	1.142	1.844	1.120	2.226	1.240	2.437	1.268	2.377
MST	1.007	1.161	1.014	1.170	1.020	1.183	1.025	1.208

IC3	$\lvert N\rvert = 4$		$\lvert N\rvert = 5$		$\lvert N\rvert = 6$		$\lvert N\rvert = 7$	
(cost)	ave	max	ave	max	ave	max	ave	max
MST	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
SPT	1.207	2.106	1.283	2.605	1.342	2.775	1.381	2.273
LDT	1.006	1.139	1.010	1.142	1.011	1.143	1.012	1.143
ORT	1.006	1.139	1.012	1.140	1.016	1.199	1.019	1.722

Table 6: 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 $|N| = 5$; 0.0003 for $|N| = 6$; and 0.0005 for $|N| = 7$.

Table 5 yields what appears to be worst case in terms of the optimality of LDTs. The IC2 technology under 7 pins gives an average value within 2.3% of ORT with a 95% confidence interval of 2.0% to 2.6%. From Table 6, we see that the Elmore-based LDT constructions are very close to optimal for the IC3 technology: for 7 pins technology they are within 0.5% of ORT delay on average. Note that under the IC3, the performance of the MST construction improves significantly, while the performance of SPT is relatively worse. By contrast, the LDT algorithm produces very good results for each of the three technologies, as is expected since it optimizes Elmore delay directly.

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

Table 7 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, due to Alpert et al., is a recent 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.[5]. All delays in Table 7 are calculated using the Two-Pole simulator. 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. 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.

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. Thus, we have addressed the issue of the accuracy and *fidelity* of the Elmore [8] and Two-Pole [21] delay models by comparing the rankings of tree topologies by these models with rankings by the SPICE simulator. Our studies indicate that algorithms which minimize the Elmore and Two-Pole estimators should also effectively minimize actual delay. We have also used the branch-and-bound BBORT method to determine *optimal* routing trees for a given monotonic delay function.

To achieve a more practical approach, we have proposed the greedy *Low Delay Tree* (LDT) heuristic. LDT can be implemented using any given model of delay, and because of the fidelity shown by 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.

The 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 [3] 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 [3]. 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.

Interestingly, since 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 tool implementations [4] [17]. 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*, Tech. Rep. CSD-TR-920051, Computer Science Department, UCLA, 1992.

[2] 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.

[3] K. D. Boese, A. B. Kahng, and G. Robins, *High-Performance Routing Trees With Identified Critical Sinks*, Tech. Rep. CS-92-37, Computer Science Department, University of Virginia, November 1992.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

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

[14] 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.

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

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

[17] B. Ramkumar and P. Banerjee, *ProperCAD: A Portable Object-Oriented Parallel Environment for VLSI CAD*, in Proc. IEEE Intl. Conf. on Computer Design, 1992.

[18] 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.

[19] K. C. Saraswat and F. Mohammadi, *Effects of Scaling of Interconnections on the Time Delay of VLSI Circuts*, IEEE Journal of Solid State Circuits, SC-17 (1982), pp. 275–280.

[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, 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.

Appendix

IC1				
1-25	26-50	51-75	76-100	101-125
1.000	1.497	2.055	2.929	3.792
1.017	1.536	2.121	2.974	3.825
1.060	1.546	2.121	3.020	3.850
1.090	1.577	2.158	3.027	3.882
1.108	1.612	2.197	3.051	3.923
1.124	1.628	2.217	3.103	3.975
1.161	1.639	2.259	3.136	4.047
1.197	1.639	2.318	3.155	4.075
1.215	1.685	2.343	3.172	4.132
1.225	1.685	2.421	3.190	4.207
1.234	1.695	2.450	3.221	4.292
1.234	1.711	2.500	3.301	4.337
1.268	1.729	2.525	3.319	4.387
1.295	1.762	2.560	3.341	4.457
1.314	1.780	2.626	3.373	4.524
1.332	1.780	2.626	3.396	4.587
1.340	1.820	2.652	3.468	4.702
1.395	1.836	2.699	3.488	4.767
1.405	1.880	2.719	3.532	4.848
1.424	1.903	2.729	3.560	4.913
1.433	1.919	2.774	3.588	5.007
1.433	1.946	2.829	3.624	5.302
1.446	1.991	2.885	3.651	5.564
1.453	2.022	2.912	3.708	5.826
1.481	2.031	2.919	3.736	6.068

IC2				
1-25	26-50	51-75	76-100	101-125
1.000	1.376	1.732	2.141	2.605
1.038	1.391	1.743	2.157	2.622
1.065	1.395	1.766	2.167	2.652
1.091	1.411	1.777	2.185	2.677
1.106	1.419	1.798	2.218	2.704
1.124	1.427	1.821	2.236	2.742
1.126	1.440	1.833	2.256	2.776
1.140	1.458	1.846	2.268	2.810
1.157	1.472	1.861	2.284	2.848
1.166	1.484	1.899	2.296	2.869
1.179	1.499	1.911	2.317	2.894
1.186	1.508	1.937	2.331	2.917
1.199	1.525	1.943	2.346	2.978
1.203	1.546	1.957	2.363	3.019
1.208	1.565	1.973	2.377	3.047
1.218	1.581	1.987	2.395	3.097
1.265	1.595	1.998	2.410	3.153
1.278	1.603	2.026	2.438	3.188
1.295	1.622	2.040	2.465	3.220
1.308	1.641	2.057	2.474	3.282
1.315	1.650	2.070	2.493	3.330
1.320	1.667	2.080	2.511	3.458
1.340	1.682	2.086	2.543	3.574
1.349	1.695	2.098	2.564	3.758
1.359	1.709	2.123	2.588	3.882

Table 8: SPICE performance ratios of all 125 topologies for $|N| = 5$ using IC1 and IC2 technology parameters. All values are averaged over 20 random sets of pin locations.

IC3				
1-25	26-50	51-75	76-100	101-125
1.000	1.142	1.213	1.281	1.351
1.016	1.145	1.216	1.286	1.355
1.030	1.147	1.219	1.289	1.359
1.042	1.150	1.221	1.292	1.361
1.052	1.154	1.224	1.296	1.364
1.059	1.155	1.228	1.301	1.367
1.063	1.160	1.231	1.302	1.370
1.068	1.162	1.234	1.307	1.374
1.076	1.167	1.236	1.309	1.377
1.082	1.170	1.238	1.313	1.379
1.089	1.173	1.239	1.316	1.382
1.093	1.175	1.241	1.319	1.385
1.097	1.179	1.245	1.321	1.386
1.101	1.180	1.247	1.323	1.390
1.107	1.183	1.250	1.324	1.394
1.110	1.186	1.251	1.326	1.399
1.113	1.188	1.253	1.328	1.402
1.116	1.191	1.255	1.330	1.407
1.119	1.193	1.258	1.333	1.411
1.123	1.195	1.262	1.337	1.415
1.125	1.198	1.264	1.339	1.420
1.127	1.200	1.268	1.341	1.428
1.133	1.204	1.271	1.342	1.433
1.136	1.208	1.274	1.345	1.442
1.140	1.211	1.278	1.348	1.447

Table 9: SPICE performance ratios of all 125 topologies for $|N| = 5$ using IC3 technology parameters. All values are averaged over 20 random sets of pin locations.