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

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 delays. We study the accuracy and fidelity of easily-computed delay models with respect to detailed simulation (e.g., SPICE-computed delays). We show that Elmore delay minimization is a high-fidelity interconnect objective for IC interconnect technologies, and 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 branch-and-bound search. Experimental results show that the LDT heuristic approximates ORTs very accurately: for nets with up to 7 pins, LDT trees have a maximum sink delay within 2.3% of optimum on average. Moreover, compared with minimum spanning tree constructions, the LDT achieves average reductions in delay of up to 35% depending on the net size.*

1 Introduction

Over the last several decades, interconnection delay has had an increasing impact on circuit speed, and now contributes up to 70% of the clock cycle in the design of dense, high-performance circuits [16]. Thus, 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., [5] [10] [11].

Once a module placement has been fixed, timing-driven interconnection algorithms are required. 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., the static timing approach of Dunlop et al. [6] and the hierarchical approach of Jackson, Kuh and Marek-Sadowska [8]. Methods for designing interconnection trees include A* heuristic search by Prastjutrakul and Kubitz [12] using the Elmore delay formula; simultaneous tree cost and radius minimization by Cong et al. [3] and by Alpert et al. [1]; use of rectilinear Steiner arborescences by Cong et al. [4]; and "critical sink" routing by Boese et al. [2].

Previous methods have often relied on simple geometric delay *abstractions*, e.g., tree cost or tree radius. In contrast, we begin our work from "first principles"; we seek a model of delay that is both algorithmically tractable and has high *fidelity* with respect to SPICE computed delays. We study the Elmore delay formula [7] and find it to be a high-fidelity routing objective.[1] Because exhaustive enumeration of all possible routing topologies is infeasible for minimizing Elmore delay, we complement our studies of fidelity with a practical, *greedy* construction (the Low-Delay Tree, or LDT heuristic). In our simulations, the Elmore-based LDT solutions closely match (to within 2.3%, on average) the delays of Elmore-optimal spanning-tree solutions. LDT routings also give delay reductions of up to 35% compared to traditional minimum spanning trees depending on the size of the net.

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 an "optimal routing tree" (ORT), 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 its *tree delay* $t(T(N)) = \max_{i=1}^{k} t(n_i)$ is minimized.

The specific routing tree that solves the ORT problem will depend on the method used to estimate delay. Generally, the circuit simulator SPICE is regarded as

*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.

[1] Kim et al. [9] have similarly shown Elmore delay to be a consistent objective by plotting Elmore versus HSPICE delay for different layout solutions for a single circuit.

the best tool for obtaining precise estimates of interconnect delay. However, the computation times required by SPICE are very large. The linear delay approximation has been used in the past (e.g., [3] and [16]), but is known to be inaccurate. Thus, the Elmore delay formula [7] and the "Two-Pole" approximation developed by Zhou et al. [17] are both of interest, because both are more accurate than linear delay and also require 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. 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 extend t_{ED} to trees by defining: $t_{ED}(T(N))= \max_{i=1}^{k} t_{ED}(n_i)$. The delay $t_{ED}(n_i)$ can be calculated in $O(k)$ time, as noted by [15]. If r_{e_j} and c_{e_j} are proportional to the length of e_j, then $t_{ED}(n_i)$ is quadratic in the length of the n_0-n_i path and linear in total wirelength (which is proportional to C_{n_0}).

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: for larger r_d, the optimal routing tree is a minimum cost spanning tree, while for smaller r_d, the ORT will tend to possess a "star" topology. Typical relative magnitudes of r_d are large for current generation CMOS, but decrease in submicron CMOS IC and MCM substrate interconnects.

3 Accuracy and Fidelity

In choosing a delay simulator, one traditionally measures the *accuracy* of the available choices. Thus, our first studies measure how close linear, Elmore, and Two-Pole delay estimates are to SPICE3e2 delay in a net.[2] Table 1 shows the average ratio between SPICE delay and both the Elmore and Two-Pole models; it also shows the consistency of this ratio in terms of its standard deviation. For each net size, the results are computed for 100 random nets connected using the minimum cost spanning tree (MST) construction, which gives relatively good solutions.

[2] Our SPICE3e2 delay model uses constant resistance and capacitance values per unit of interconnect. The root of the tree is driven by a resistor connected to the source, and each sink is modeled by an inverter. We use a step function to model the input signal. The technology parameters we use are representative of a typical 0.8μ CMOS process, with driver resistance 100 Ω; wire resistance 0.03 $\Omega/\mu m$; wire capacitance 0.352 $fF/\mu m$; wire inductance 492 $fH/\mu m$; sink loading capacitance 15.3 fF; and layout area 10^2 mm^2.

	$\|N\| = 4$		$\|N\| = 7$	
	average	standard deviation	average	standard deviation
SPICE/Elmore	1.51	0.19	1.31	0.13
SPICE/2-Pole	0.64	0.09	0.60	0.06

Table 1: Average and standard deviation of the ratio between "actual" SPICE3e2 delay and estimated MST delay over 100 random nets chosen from a uniform distribution over the layout area.

Table 1 indicates that neither the Elmore nor Two-Pole delay models produce highly accurate delay estimates. Only for 7-pin nets 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 we have tested. However, the Elmore and Two-Pole delay estimators are very consistent, with small standard deviations ranging from 10% to 14% of the average ratio.

Precise accuracy is often not required of delay estimates used to construct routing trees. In practice, we require good estimators to have a high degree of of *fidelity* (i.e, the degree to which an optimal or near-optimal solution according to the estimator will also be nearly optimal according to actual delay). To this end, we have defined 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.

Topologies	$\|N\| =$	Linear vs SPICE		Elmore vs SPICE		2-Pole vs SPICE	
		4	5	4	5	4	5
All		1.33	8.69	0.82	4.75	0.44	2.93
Best		2.05	6.25	0.70	1.45	0.10	0.45
5 Best		1.47	6.24	1.05	3.30	0.53	1.24

Table 2: Average difference in topology rankings for linear, Elmore and 2-pole models compared to SPICE3e2. The sample consists of 20 random nets of each cardinality. The total number of topologies for each net is 16 when $|N| = 4$ and 125 when $|N| = 5$.

Table 2 assesses the fidelity to SPICE of the linear, Elmore, and Two-Pole delay estimators for nets of size 4 and 5. We report the average difference in ranking over all topologies; the average rank difference for the topology with lowest estimated (i.e., non-SPICE) delay; and the average difference for the five topologies which have lowest estimated delay. Our results show that Elmore delay has high fidelity, particularly for the "best" topology under Elmore delay: for 5-pin nets, this topology has average distance of 1.5 from its SPICE ranking, compared to 6.3 for the best topology under linear delay.

The average difference of 1.5 positions between the best Elmore and SPICE topologies for $|N| = 5$ leads to approximately a 5% penalty in SPICE-computed delay. This can be seen in Table 3, which shows the average drop-off in SPICE3e2 delay by topology rank, when compared with optimal delay.

Table 2 indicates that the Two-Pole simulator has

somewhat better fidelity than Elmore delay. However, because of its speed in calculation, Elmore delay is more practical for searching over tree topologies.

1-5	6-10	11-15	16-20	21-25
1.000	1.124	1.179	1.218	1.315
1.038	1.126	1.186	1.265	1.320
1.065	1.140	1.199	1.278	1.340
1.091	1.157	1.203	1.295	1.349
1.106	1.166	1.208	1.308	1.359

Table 3: Average SPICE3e2 delay ratios of best 25 topologies for $|N| = 5$. Values are normalized to the best delay of any topology and averaged over 20 random nets.

4 Near-Optimal Routing Trees

We can solve the ORT problem *optimally* for a given delay model using backtracking enumeration of 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 prune the search and backtrack to select a different edge at the previous step. We call this optimal method *Branch-and-Bound ORT* (BBORT) search. BBORT finds the optimal-delay tree for any *monotone* delay function, where the tree delay does not decrease with the addition of a new edge. Note that despite BBORT's pruning of the solution space, its worst-case time complexity is still exponential.

To avoid the exponential running time of BBORT, we propose the following greedy heuristic to approximate ORTs. Our method is analogous to Prim's minimum spanning tree construction [14]: 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. Figure 1 gives pseudo-code for this *Low Delay Tree* (LDT) heuristic.

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 1: The Low Delay Tree heuristic.

The LDT heuristic generalizes the Elmore Routing Tree algorithm of Boese et al.[2] to any given delay model. For Elmore delay, LDT can easily be implemented in $O(k^3)$ time using the following observation[3]: if a new tree edge incident to sink $v \in V$ (Line 3 of Figure 1) minimizes the maximum delay, 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 1, we can update the shortest "outside connections" for all $v \in V$ (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, and each pass through the while loop requires $O(k^2)$ time, yielding the overall complexity of $O(k^3)$. In practice this is very reasonable, since k is small.

5 Experimental Results

We have implemented both BBORT and LDT for Elmore delay using C in the UNIX/Sun environment. We have run trials on sets of 500 uniformly distributed nets for each of several net sizes, using the IC technology parameters described in footnote 2. Table 4 compares Elmore delays of these constructions with delays of minimum spanning trees (MST) and shortest path trees (SPT)[4]. Delay for each tree is normalized to the ORT delay over the same net, and cost is normalized to MST cost.

We see that LDTs on 7 pins have an average maximum Elmore delay only 2.3% greater than optimal, while MSTs have average tree delay 38% greater than optimal. For smaller nets, LDTs are even closer to optimal: for nets with 4 pins, LDT delays average within 0.3% of optimal. Our confidence in the average difference computed between LDTs and ORTs is very high: for instance, the 2.3% difference obtained for 7 pins has a standard error of 0.14%, indicating a 95% confidence interval between 2.0% and 2.6% (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 11.4% for 4-pin nets and 16.4% for 7-pin nets. The high performance of LDTs is achieved with an average wirelength penalty of from 10.3% for 4-pin nets to 16.2% for 7-pin nets when compared to MST constructions.

Table 5 compares delays in Elmore-based LDTs with those of the MST and AHHK [1] constructions for nets of up to 17 pins using the same IC parameters. All delays in the table are calculated using the Two-Pole simulator. The AHHK algorithm of Alpert et al. is a recent cost-radius tradeoff construction which yields less tree cost (and signal delay) for given tree radius bounds than the method of [3]. 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 17-pin nets, the LDT construction reduces average sink delay by 35.4% compared to MSTs and by 6.2% compared to AHHK trees.

[3] The observation assumes constant loading capacitances, unit resistances, and unit capacitances.

[4] The SPT construction is the tree which minimizes cost subject to each source/sink path having minimum length, similar to the A-tree construction of [4].

	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
Delay	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
	$\|N\| = 4$		$\|N\| = 5$		$\|N\| = 7$	
Cost	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 4: Average and maximum Elmore delays and tree costs over 500 random nets. Cost values are normalized to MST cost and delays are normalized to the (Elmore-based) ORT delay. Standard errors for average LDT-Elmore delay are 0.0006 for $|N| = 4$; 0.0010 for $|N| = 5$; and 0.0014 for $|N| = 7$.

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

Table 5: Two-Pole simulator delays comparing LDT with MST and AHHK trees on nets with up to 17 pins. Averages in each column are taken over 500 random nets.

6 Conclusions

We have addressed the issue of the accuracy and *fidelity* of the Elmore [7] and Two-Pole [17] delay models by comparing their rankings of tree topologies with rankings by the SPICE3e2 simulator. Our studies indicate that algorithms which minimize the Elmore and Two-Pole delay estimates should also effectively minimize actual delay. We have also described a branch-and-bound BBORT method to determine *optimal* routing trees for any given monotone 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, LDT achieves reductions in delay of up to 35% (depending on the net size) over the MST routing, as measured by the Two-Pole simulator. Significant reductions are also achieved compared to AHHK- [1] and SPT-based routings.

LDT 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 the closest point in any existing tree edge, possibly inducing a Steiner point. Simulation results in [2] indicate that the SLDT algorithm using Elmore delay is also highly effective. LDT can also be generalized to "critical-sink routing" by modifying the objective function in the LDT and SLDT algorithms to minimize delay at prescribed critical sinks [2]. Furthermore, our constructions can be adapted to average tree delay (i.e., sum of delays to all sinks) or any other well-behaved delay function.

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*, Proc. ISCAS, 1993, pp. 1869-1872.

[2] K. D. Boese, A. B. Kahng, and G. Robins, *High-Performance Routing Trees With Identified Critical Sinks*, Proc. DAC, 1993.

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

[4] J. Cong, K. S. Leung, and D. Zhou, *Performance-Driven Interconnect Design Based on Distributed RC Delay Model*, Proc. DAC, 1993.

[5] 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*, Proc. DAC, 1990, pp. 84–89.

[6] 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. DAC, 1984, pp. 133–136.

[7] 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.

[8] M. A. B. Jackson, E. S. Kuh, and M. Marek-Sadowska, *Timing-Driven Routing for Building Block Layout*, Proc. ISCAS, 1987, pp. 518–519.

[9] S. Kim, R. M. Owens and M. J. Irwin, *Experiments with a Performance Driven Module Generator* Proc. DAC, 1992, pp. 687–690.

[10] I. Lin and D. H. C. Du, *Performance-Driven Constructive Placement*, Proc. DAC, 1990, pp. 103–106.

[11] M. Marek-Sadowska and S. P. Lin, *Timing Driven Placement*, Proc. ICCAD, 1989, pp. 94–97.

[12] S. Prasitjutrakul and W. J. Kubitz, *A Timing-Driven Global Router for Custom Chip Design*, Proc. ICCAD, 1990, pp. 48–51.

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

[14] A. Prim, *Shortest Connecting Networks and Some Generalizations*, Bell Sys. Tech J. 36(1957), pp. 1389-1401.

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

[16] S. Sutanthavibul and E. Shragowitz, *An Adaptive Timing-Driven Layout for High Speed VLSI*, Proc. DAC, 1990, pp. 90–95.

[17] 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.