Near-Optimal Critical Sink Routing Tree Constructions

Kenneth D. Boese, *Member, IEEE*, Andrew B. Kahng, *Associate Member, IEEE*, Bernard A. McCoy, *Member, IEEE*, and Gabriel Robins, *Member, IEEE*

***Abstract*—We present *critical-sink routing tree* (CSRT) constructions which exploit available critical-path information to yield high-performance routing trees. Our CS-Steiner and "global slack removal" algorithms together modify traditional Steiner tree constructions to optimize signal delay at identified critical sinks. We further propose an iterative *Elmore routing tree* (ERT) construction which optimizes Elmore delay *directly*, as opposed to heuristically abstracting linear or Elmore delay as in previous approaches. Extensive timing simulations on industry IC and MCM interconnect parameters show that our methods yield trees that significantly improve (by averages of up to 67%) over minimum Steiner routings in terms of delays to identified critical sinks. ERT's also serve as generic high-performance routing trees when no critical sink is specified: for 8-sink nets in standard IC (MCM) technology, we improve average sink delay by 19% (62%) and maximum sink delay by 22% (52%) over the minimum Steiner routing. These approaches provide simple, basic advances over existing performance-driven routing tree constructions. Our results are complemented by a detailed analysis of the accuracy and *fidelity* of the Elmore delay approximation; we also *exactly* assess the suboptimality of our heuristic tree constructions. In achieving the latter result, we develop a new characterization of Elmore-optimal routing trees, as well as a decomposition theorem for optimal Steiner trees, which are of independent interest.**

I. Introduction

DUE to the scaling of VLSI technology, interconnection delay has become a dominant concern in the design of complex, high-performance circuits [13], [34]. Performance-driven layout design has thus become an active area of research over the past several years. In this paper, we develop a new *critical-sink* problem formulation and new solutions for performance-driven routing tree design.

For a given signal net, the typical goal of performance-driven routing is to minimize average or maximum source-sink delay. Much early work implicitly equates optimal routing with minimum-cost Steiner routing. For example, [14] used static timing analysis to yield net priorities, so that the highest-priority nets may be routed by minimum Steiner trees, leaving lower-priority nets to subsequently encounter blockages. References [21], [28] have given approaches which are tuned to building-block layout and allow prescribed upper bounds on individual source-sink delays; the former work also incorporates a hierarchy-based net ordering. For minimum Steiner tree routing, the 1-Steiner method [22] is the best-performing heuristic, and we therefore use it as a basis for comparison below.[1]

Reference [9] proposed a heuristic which simultaneously considered both the *cost* (total edge length) and the *radius* (longest source-sink path length) of the routing tree. A more general formulation was given in [10], wherein a parameter ϵ guides the tradeoff between cost and radius minimization; the same authors in [10] proposed the "provably good" BRBC (bounded-radius, bounded-cost) algorithm, which affords both cost and radius simultaneously within *constant* factors of optimal. The BRBC method and works of [3], [23] all achieve a smooth cost-radius tradeoff via the same basic idea: 1) make a depth-first traversal of the minimum spanning tree over the signal net, and 2) if the accumulated path length from the source to some sink becomes too large, modify the tree to reduce that particular source-sink path length. The cost-radius tradeoff may also be viewed as one between competing minimum spanning tree (MST) (or minimum-cost Steiner tree) and shortest-path tree (SPT) constructions. Using this perspective, [1] recently proposed the AHHK algorithm, which achieves a direct MST-SPT tradeoff. Finally, [11] proposed the use of rectilinear Steiner arborescences [30], or A-trees; these are essentially minimum-cost SPT's with Steiner points allowed. The delay performance of the AHHK algorithm is superior to that of the BRBC or A-tree constructions [1], and thus below we use AHHK as another basis of comparison with our new methods.

A. Motivations for Critical-Sink Routing

In performance-driven layout for cell-based designs, timing-critical paths are determined by static timing analysis, and modules in these paths are then placed close together (see, e.g., [13], [18], [20], [26], [27], [34]). The static timing analysis thus *iteratively* drives changes within both the module placement and the global routing phases. Our contribution stems from carefully considering routing tree constructions within this overall performance-driven layout process.

Manuscript received August 26, 1993; revised July 12, 1994.

This work was supported in part by a GTE Graduate Fellowship, in part by the National Science Foundation under contracts MIP-9110696, in part by the National Science Foundation Young Investigator Award under contracts MIP-9257982 and MIP-9457412, in part by ARO under contracts DAAK-70-92-K-0001, and DAAL-03-92-G-0050, and in part by a Packard Foundation Fellowship. This paper was recommended by Associate Editor M. Sarrafzadeh.

K. D. Boese is with Cadence Design Systems, San Jose, CA 95134 USA.

A. B. Kahng is with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90024-1596 USA.

B. A. McCoy is with Nu-Mega Technologies, Nashua, NH 03060–7780 USA.

G. Robins is with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442 USA.

IEEE Log Number 9413600.

[1]Recent studies by [4], using the optimal Steiner code of J. Salowe, show that 1-Steiner is within 0.25% of optimal on average.

In general, existing performance-driven placement algorithms may be classified as either *net-oriented* or *path-oriented. Net-oriented* placement typically uses centroid-connected star cost [33], probabilistic estimates of Steiner tree cost [20], minimum spanning tree cost [13] or the bounding box semiperimeter [27] to estimate wire capacitance and signal delay for a multi-terminal net. From this information, critical timing paths between primary inputs and primary outputs are computed, after which module placements are updated to reduce these "net-based" objectives for signal nets along the critical paths. By contrast, *path-oriented* placement considers delay between the source and a particular *critical sink* of a multi-terminal net. The critical sink is typically determined via timing analysis using known module delays and estimated path delays. For example, [26] used a linear delay approximation so that their method updates the module placement to reduce the rectilinear distance between sources and critical sinks. Other path-oriented methodologies include those of [18], [35].

If a timing-critical path passes through a given net, the path-oriented approach can provide an explicit bound on delay at that net's critical sink. While the net-oriented approach may arguably provide only implicit routing constraints, it is still easy to identify critical sinks after the timing analysis has been performed, or *a priori* by finding paths in the design that contain more module delays. This reveals a "placement-routing mismatch:" the performance-driven routing constructions reviewed above generally address *net-specific* objectives (min cost, min radius, cost-radius tradeoffs, etc.) and do not exploit the critical-path information that is available during iterative performance-driven layout. As a consequence, designers cannot realize the full benefit of high-quality timing-driven module placements. With this in mind, our work develops new high-performance routing tree constructions which *directly* exploit available critical-path timing information.

B. The Critical-Sink Routing Tree Problem

A *signal net* N consists of a set of pin locations $\{n_0, n_1, \ldots, n_k\}$ in the Manhattan plane, which are to be connected by a *routing tree* $T(N)$. We use n_0 to denote the *source*, with the n_i $(1 \leq i \leq k)$ denoting *sinks*. The *cost* of an edge e_{ij} in $T(N)$, denoted by d_{ij}, is the Manhattan distance between the endpoints n_i and n_j of the edge. The cost of the tree $T(N)$ is simply the sum of its edge costs. In a given routing tree $T(N)$, the signal delay between two terminals n_i and n_j is denoted by $t(n_i, n_j)$; the shorthand notation $t(n_i)$ indicates the delay from the source to the sink n_i. Finally, we allow each n_i to have an associated *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 sink delays:

Critical-Sink Routing Tree (CSRT) Problem: Given a signal net $N = \{n_0, n_1, \ldots, n_k\} \subset \Re^2$ 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.

This CSRT problem formulation is quite general, and easily captures traditional performance-driven routing tree objectives: 1) *average delay* to all sinks is minimized by using all α_i = some positive constant, then taking the L_1 sum of the weighted delays; and 2) *maximum delay* to any sink is minimized by using all α_i = some positive constant, then taking the L_∞ sum of the weighted delays. In the discussion below, we will concentrate on the simple yet realistic case where *exactly one* critical sink, denoted by n_c, has been identified. In other words, we assume that $\alpha_c > 0$ and that all other $\alpha_i = 0$. Our methods may be generalized to the case where a small number of critical sinks is specified.

The remaining discussion is organized as follows: Section II discusses the appropriate choice of a delay measure to guide the routing tree design, and derives motivating observations from analysis of the Elmore approximation for signal delay in distributed RC trees. Section III then presents our two main classes of CSRT algorithms. We first describe the *CS-Steiner* method, which perturbs an existing Steiner tree construction to account for the presence of identified critical sinks. We then propose an efficient class of *Elmore routing tree* (ERT) constructions which not only yield good CSRT solutions, but are also the first methods to optimize Elmore delay *directly* without any of the abstractions implicit in previous routing objectives. Section III also describes the extension of the ERT approach to net-dependent routing objectives. Experimental results are presented in Section IV, where we compare delays at critical sinks in our heuristic tree topologies with analogous delays obtained using the best-performing minimum Steiner tree heuristic [22] and the AHHK routing [1]. Our methods prove extremely effective, obtaining up to an *average* 69% reduction in signal delay to identified critical sinks in 8-sink nets. The ERT approach also yields *generic* high-performance routing trees when all sinks are equally critical: for 9-pin nets in 1.2μ CMOS IC (MCM) technology, we improve average sink delay by 19% (62%) and maximum delay by 22% (52%) over the minimum Steiner routing. We thus obtain a significant advance over the existing performance-driven routing tree constructions in the literature, including such recent works as [1], [10], [28]. Our results are complemented by a detailed analysis of the accuracy and *fidelity* of the Elmore delay approximation, and we furthermore provide *exact assessments versus optimal* for our heuristic tree constructions. To determine the latter data, we have developed a new theoretical characterization of Elmore-optimal routing trees, as well as a decomposition theorem for (Elmore-) optimal Steiner trees, which are of independent interest.

II. On Delay Approximations and Tree Design Objectives

For arbitrary signal nets N, the appropriate objective to use in *efficiently* constructing "high-performance routing trees" has not yet been established. In this section, we first consider necessary qualities for a delay approximation that is to be used in routing tree design. By studying both the relative accuracies and the relative *fidelities* of linear, distributed RC, distributed RCL, and SPICE-computed delay approximations, we demonstrate that the Elmore distributed RC delay approximation is of surprisingly high fidelity with respect to SPICE3e2. From Elmore's simple formula (i.e., the first moment of the impulse response in a distributed RC tree), we then develop revealing

TABLE I
TECHNOLOGY PARAMETERS FOR THREE CMOS IC TECHNOLOGIES AND ONE MCM TECHNOLOGY. PARASITICS AND SPICE SIMULATION DECKS FOR THE IC1 AND IC2 TECHNOLOGIES ARE PROVIDED BY MOSIS; IC3 PARASITICS ARE COURTESY OF MCNC. THE DRIVER RESISTANCES (R_D) AND SINK LOADING CAPACITANCES ARE DERIVED FOR MINIMUM-SIZE TRANSISTORS. NOTE THAT INDUCTANCE VALUES FOR IC1–IC3 ARE SET TO 1×10^{-5} fH/μm (EFFECTIVELY ZERO) BECAUSE THEY WERE NOT PROVIDED BY MOSIS/MCNC, AND BECAUSE A NONZERO INDUCTANCE IS REQUIRED BY THE TWO-POLE SIMULATOR (SEE FOOTNOTE 4). MCM INTERCONNECT PARASITICS ARE COURTESY OF PROFESSOR W. W.-M. DAI OF THE UNIVERSITY OF CALIFORNIA, SANTA CRUZ, AND CORRESPOND TO DATA PROVIDED BY AT&T MICROELECTRONICS DIVISION

Name	IC1	IC2	IC3	MCM
Technology	2.0 μm CMOS	1.2 μm CMOS	0.5 μm CMOS	MCM
r_d	164.0 Ω	212.1 Ω	270.0 Ω	25.0 Ω
unit wire resistance	0.033 $\Omega/\mu m$	0.073 $\Omega/\mu m$	0.112 $\Omega/\mu m$	0.008 $\Omega/\mu m$
unit wire capacitance	0.234 $fF/\mu m$	0.083 $fF/\mu m$	0.039 $fF/\mu m$	0.06 $fF/\mu m$
unit wire inductance	1x10^{-5} $fH/\mu m$	1x10^{-5}$fH/\mu m$	1x10^{-5}$fH/\mu m$	380 $fH/\mu m$
loading capacitance	5.7 fF	7.06 fF	1.0 fF	1000 fF
resistance ratio (x $10^6 \mu m$)	0.0050	0.0029	0.0024	0.0031
chip size	1x1 cm^2	1x1 cm^2	1x1 cm^2	10x10 cm^2

intuitions regarding the "correct" objective for critical-sink routing tree design.

A. Accuracy and Fidelity of Delay Approximations

Ideally, a routing algorithm will compute and optimize signal delays according to a detailed circuit simulation, such as that provided by SPICE. Since the computation times required by SPICE are prohibitive for routing tree construction, simpler delay approximations must be used. For example, the traditional minimum-cost Steiner tree objective, in addition to minimizing wiring area, corresponds to a *lumped-capacitance* model (i.e., signal delay is proportional to total tree capacitance, which is proportional to tree cost). In [9], [10], [34], the *linear* delay approximation is used; sink delays are thus proportional to source-sink path lengths, and a minimum-radius criterion is obtained.

Such simple delay approximations are known to be inaccurate as technology scales, e.g., smaller wire geometries imply that resistive effects of the interconnect become more dominant, particularly in relation to driver on-resistance (see the discussion below of "resistance ratio" effects, and note the four technology characterizations in Table I). Furthermore, greater system speeds and layout areas may expose inductive effects on delay. Given these considerations, distributed RC delay approximations (e.g., that of Elmore [15]) or distributed RCL delay approximations (e.g., the "Two-Pole" simulator of [38]) are of interest, since they are more accurate than linear or lumped-capacitance approximations while requiring less computation time than SPICE.

Elmore delay in an RC tree [15], [32], [36] is defined as follows: Given routing tree $T(N)$ rooted at the source n_0, let e_v denote the edge from node v to its parent in $T(N)$. The resistance and capacitance of edge e_v are denoted by r_{e_v} and c_{e_v}, respectively. Let T_v denote the subtree of T rooted at v, and let c_v denote the sink capacitance of v ($c_v = 0$ if v is a Steiner node). We use C_v to denote the *tree capacitance* of T_v, namely the sum of sink and edge capacitances in T_v. Using this notation, the Elmore delay along edge e_v is equal to $r_{e_v}(\frac{c_{e_v}}{2} + C_v)$. Let r_d denote the output driver resistance at the net's source. Then Elmore delay $t_{\text{ED}}(n_i)$ at sink n_i is

$$t_{\text{ED}}(n_i) = r_d C_{n_0} + \sum_{e_v \in \text{path}(n_0, n_i)} r_{e_v}\left(\frac{c_{e_v}}{2} + C_v\right). \quad (1)$$

Although Elmore delay has a compact definition and can be quickly computed[2] it does not capture all of the factors that account for delay. For example, the Two-Pole distributed RCL simulator [38] considers inductive effects; according to [5], [38], its moment-based methodology is intermediate between SPICE and Elmore delay in both accuracy and computational efficiency.

Accuracy: In choosing a delay simulator, one traditionally measures *accuracy*, which may vary with the circuit technology and the specifics of a net (for instance, the number of pins it contains, or the size of its bounding box). Tables II and III indicate the accuracy of the linear, Elmore and Two-Pole models for each of the interconnect technologies described in Table I. For each of the three estimators, the tables give the average ratio of SPICE delay to the estimated delay, and also show the consistency of this ratio in terms of its standard deviation.[3] In Table II, delay is calculated for a single random "critical" sink; in Table III, delay is measured as the maximum delay at any sink in the net. For each net size, the results are averaged over 100 random nets with pin locations chosen from a uniform distribution over the routing area; each net is connected using the minimum cost spanning tree (MST) construction. We use MST's rather than random tree

[2]Elmore delay can be evaluated at *all* sinks in $O(k)$ time, as noted in [32]. The calculation uses two depth-first traversals: 1) to compute the delay along each edge and 2) to sum up the delays along each source-sink path.

[3]Our SPICE3e2 delay modeling uses constant unit resistance and capacitance values. The root of the routing tree is driven by a resistor connected to the source. For the Two-Pole and SPICE simulators, every interconnect segment is broken into uniform segments, each at most 1/100th the length of the layout dimension, connected in series. To model sink loads, we use pure capacitive loads derived using minimum-size transistors. For all simulators, we have used the 50% rise time delay criterion, and we have measured both average sink delay and maximum sink delay. For the Two-Pole and SPICE simulators we have used time steps of 0.005 ns for the IC technologies and 0.05 ns for MCM.

We have found our results to be qualitatively independent of methodological choices (e.g., 50% rise time instead of 90% rise time as a delay criterion). However, many reasonable alternative simulation methodologies were possible. For instance, Elmore delay does not intrinsically correspond to any delay time (it is simply the first moment of an impulse response), but can be said to correspond to a 63% delay criterion since RC/2 is the coefficient of s in the system transfer function $H(s)$ of a distributed RC line. On the other hand, the nature of the Two-Pole approximation makes it more suited to a 90% rise time criterion [38]. Other inconsistencies: while SPICE can model active devices as loads, the Two-Pole simulator can only handle "equivalent" sink capacitances; while SPICE and Two-Pole can model series inductance (for MCM interconnect), Elmore delay is solely a distributed RC model—indeed, the list of incomparable variables seems endless.

TABLE II

ACCURACY OF THE LINEAR, ELMORE AND TWO-POLE ESTIMATORS FOR CRITICAL-SINK DELAY. THIS TABLE GIVES THE AVERAGE AND STANDARD DEVIATION OF THE RATIOS BETWEEN SPICE3E2 DELAY AND ESTIMATED DELAY AT A SINGLE RANDOM CRITICAL SINK, AVERAGED OVER 100 RANDOM NETS. ALL NETS ARE CONNECTED USING MST CONSTRUCTIONS. STANDARD DEVIATIONS ARE REPORTED AS A PERCENT OF THE AVERAGE. LINEAR DELAY IS DEFINED AS THE SOURCE/SINK PATHLENGTH; BECAUSE THIS IS A DISTANCE RATHER THAN A TIME, WE DO NOT REPORT A SPICE/LINEAR "RATIO." HOWEVER, WE CAN REPORT THE STANDARD DEVIATION OF THIS QUOTIENT, SINCE IT IS INDEPENDENT OF UNITS

Accuracy of Linear, Elmore and Two-Pole Delay Estimates for Critical-Sink Delay					
		$\|N\| = 4$		$\|N\| = 7$	
	Delay Ratio	average	std dev	average	std dev
IC1	SPICE/Linear†	–	28.4%	–	32.7%
	SPICE/Elmore	0.72	13.5%	0.69	15.4%
	SPICE/2-Pole	1.27	13.5%	1.23	15.4%
	2-Pole/Elmore	0.568	0.5%	0.566	0.2%
IC2	SPICE/Linear†	–	33.9%	–	38.8%
	SPICE/Elmore	0.74	16.1%	0.70	17.8%
	SPICE/2-Pole	1.30	15.9%	1.23	17.8%
	2-Pole/Elmore	0.572	0.9%	0.568	0.5%
IC3	SPICE/Linear†	–	34.9%	–	40.3%
	SPICE/Elmore	0.78	16.0%	0.72	17.8%
	SPICE/2-Pole	1.36	15.7%	1.27	17.9%
	2-Pole/Elmore	0.574	1.4%	0.571	0.8%
MCM	SPICE/Linear†	–	57.1%	–	61.6%
	SPICE/Elmore	0.69	20.5%	0.65	25.1%
	SPICE/2-Pole	1.20	20.8%	1.14	25.2%
	2-Pole/Elmore	0.568	1.0%	0.566	0.4%

TABLE III

ACCURACY OF THE LINEAR, ELMORE AND TWO-POLE ESTIMATORS FOR MAXIMUM SINK DELAY. SEE TABLE II FOR EXPLANATORY NOTES

Accuracy of Linear, Elmore and Two-Pole Delay Estimates for Maximum Sink Delay					
		$\|N\| = 4$		$\|N\| = 7$	
	Delay Ratio	average	std dev	average	std dev
IC1	SPICE/Linear	–	11.0%	–	11.4%
	SPICE/Elmore	0.79	1.9%	0.79	1.7%
	SPICE/2-Pole	1.39	1.7%	1.39	1.7%
	2-Pole/Elmore	0.568	0.3%	0.567	0.2%
IC2	SPICE/Linear	–	12.5%	–	12.9%
	SPICE/Elmore	0.83	4.1%	0.81	2.6%
	SPICE/2-Pole	1.44	3.6%	1.42	2.4%
	2-Pole/Elmore	0.572	0.8%	0.568	0.4%
IC3	SPICE/Linear	–	13.0%	–	13.6%
	SPICE/Elmore	0.87	6.0%	0.83	3.6%
	SPICE/2-Pole	1.51	5.1%	1.46	3.4%
	2-Pole/Elmore	0.574	1.3%	0.569	0.4%
MCM	SPICE/Linear	–	25.8%	–	23.3%
	SPICE/Elmore	0.79	2.3%	0.79	2.0%
	SPICE/2-Pole	1.39	2.1%	1.39	1.9%
	2-Pole/Elmore	0.568	1.0%	0.566	0.4%

topologies so that our comparisons will be for relatively good (albeit not necessarily optimal) routing solutions. (Observe that for a 7-pin net, finding the optimal-delay routing solution by exhaustive enumeration using SPICE is not computationally feasible.)

In all cases, the ratio of SPICE to Linear Delay has the largest standard deviation; this inaccuracy in the linear approximation is not surprising. It is also reasonable to expect poor "accuracy" of the Elmore and Two-Pole approximations with respect to SPICE, if only due to the somewhat ill-defined state of delay modeling and analysis noted in Footnote 3. Indeed, based on the average ratio of SPICE to Elmore delay or to the Two-Pole simulator, neither estimator seems particularly accurate: each is generally at least 20% away from SPICE on average at the critical sink.[4] Interestingly, Table III shows the ratio between SPICE and both the Two-Pole and Elmore estimators to be very consistent when measuring *maximum* sink delay, with standard deviations within 4% for all technologies on seven-pin nets. Thus, precomputed "correction factors" would seemingly compensate for the observed inaccuracy of these estimators. However, for delay at a random critical sink, the standard deviation of the accuracy ratio is consistently above 15%. This lesser consistency perhaps indicates that the traditional net-based performance objective is more "forgiving" of errors in the delay estimate than newer path-based delay objectives.[5]

Fidelity: A key observation is that precise accuracy is *not* really required of delay estimates used to construct routing trees. In practice, we only require that an estimator have a high degree of *fidelity*, i.e, an optimal or near-optimal solution according to the estimator should 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* spanning tree topologies[6] by the given delay model, then rank the topologies again by SPICE delay, and then find the average of the absolute value of the difference between the two rankings for each topology. This measure of fidelity corresponds to a standard rank-ordering technique used in the social sciences [2]. We have run simulations to estimate this measure of fidelity for nets of size 4 and 5 using the linear and Elmore delay estimators and each of the four interconnect technologies.[7] (In this section, we show that the ratio between Elmore delay and the Two-Pole estimator of [38] is very nearly constant. As would therefore be expected, fidelity values for the Two-Pole simulator are nearly identical to those for Elmore delay, and we do not report them here.)

Tables IV and V show the fidelity to SPICE of the linear and Elmore delay estimators; the delay criterion is the 50% delay time to a given randomly-chosen critical sink in the net. We report the average difference in ranking over all topologies; the average rank difference for the topology which has lowest

[4] Recall that we assigned a near-zero inductance value for the IC technologies, since inductance parameters were not available from MOSIS/MCNC, and since the Two-Pole simulator requires a nonzero inductance. We found that this does not change our results significantly. For example, if we increase the IC1 inductance parameter to 400 f H/μm, the average Two-Pole/Elmore ratio becomes 0.566 with a standard deviation of 0.51%. Elmore delay is independent of inductance since the first moment of the impulse response in a distributed RCL tree has no L term.

[5] The small standard deviation of the accuracy values for maximum sink delay seems in part due to the rough similarity of the maximum source-sink distances over the examples studied. Note that, for example, with the MCM technology and $|N| = 7$, the average SPICE/Elmore ratio for a random sink is 0.65, whereas for the sink with greatest delay (generally furthest from the source) the ratio is 0.79. The critical-sink analysis in some sense better reveals this effect of sink distance from the source.

[6] There are $|N|^{|N|-2}$ distinct spanning tree topologies for any given net N [16].

[7] Again, we use linear delay defined to be the source/sink pathlength. This definition leads to numerous ties between topologies, and we break ties in favor of trees with lower total wirelength. Ties also occur with SPICE-computed delay because of the finite time step used; here we again break ties according to total wirelength.

TABLE IV
AVERAGE DIFFERENCE IN RANKINGS OF TOPOLOGIES, IN TERMS OF 50% DELAY TO A GIVEN RANDOM **CRITICAL SINK** IN EACH NET. THE SAMPLE CONSISTS OF 50 RANDOM NETS OF EACH CARDINALITY, AND THE 50% RISE TIME DELAY CRITERION WAS USED. THE TOTAL NUMBER OF TOPOLOGIES FOR EACH NET IS $4^{(4-2)} = 16$ FOR $|N| = 4$, AND $5^{(5-2)} = 125$ FOR $|N| = 5$

		Linear vs SPICE		Elmore vs SPICE	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	Best	2.30	16.3	0.54	5.9
	5 Best	2.52	18.1	1.02	7.2
	All	2.43	17.0	0.92	8.0
IC2	Best	2.52	19.4	0.58	6.4
	5 Best	2.66	20.2	0.99	7.2
	All	2.44	16.9	0.94	7.9
IC3	Best	2.60	19.8	0.58	5.6
	5 Best	2.68	20.9	0.93	6.5
	All	2.43	16.5	0.93	7.7
MCM	Best	3.04	24.6	0.72	5.1
	5 Best	2.81	24.4	0.89	4.7
	All	2.33	15.7	0.89	7.1

TABLE V
AVERAGE DIFFERENCE IN RANKINGS OF TOPOLOGIES IN TERMS OF **MAXIMUM** DELAY OVER ALL SINKS. THE SAMPLE CONSISTS OF 50 RANDOM NETS OF EACH CARDINALITY, AND THE 50% RISE TIME DELAY CRITERION WAS USED

		Linear vs SPICE		Elmore vs SPICE	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	Best	0.50	2.06	0.38	0.10
	5 Best	0.66	2.78	0.71	0.47
	All	0.94	7.74	0.65	1.39
IC2	Best	0.40	2.26	0.16	0.20
	5 Best	0.68	2.61	0.51	0.53
	All	0.87	7.02	0.43	1.24
IC3	Best	0.64	2.40	0.48	0.20
	5 Best	0.87	2.59	0.52	0.44
	All	1.04	6.96	0.60	1.22
MCM	Best	0.70	4.56	0.14	0.08
	5 Best	0.71	3.15	0.11	0.22
	All	1.02	7.01	0.16	0.86

TABLE VI
AVERAGE SPICE DELAY RATIOS FOR THE TOP 20 TOPOLOGIES RANKED ACCORDING TO SPICE FOR $|N| = 5$. VALUES ARE AVERAGED OVER 50 RANDOM NETS AND NORMALIZED TO THE AVERAGE DELAY OF THE BEST TOPOLOGY. ALSO INCLUDED IS THE AVERAGE RATIO FOR THE **WORST** TOPOLOGY (RANK 125)

	IC1		IC2		IC3		MCM	
Rank	CS	Max	CS	Max	CS	Max	CS	Max
1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
2	1.006	1.049	1.003	1.046	1.002	1.050	1.001	1.044
3	1.011	1.087	1.005	1.088	1.005	1.089	1.001	1.108
4	1.014	1.120	1.006	1.128	1.006	1.133	1.002	1.189
5	1.016	1.154	1.007	1.153	1.006	1.158	1.003	1.250
6	1.017	1.180	1.007	1.184	1.006	1.191	1.004	1.297
7	1.026	1.201	1.012	1.215	1.007	1.221	1.005	1.363
8	1.040	1.227	1.021	1.243	1.014	1.247	1.005	1.419
9	1.074	1.253	1.046	1.273	1.036	1.279	1.014	1.474
10	1.160	1.282	1.138	1.311	1.120	1.322	1.047	1.531
11	1.180	1.306	1.155	1.336	1.134	1.345	1.049	1.594
12	1.224	1.330	1.207	1.371	1.182	1.380	1.058	1.652
13	1.246	1.353	1.218	1.399	1.191	1.417	1.060	1.713
14	1.288	1.387	1.254	1.436	1.225	1.449	1.064	1.763
15	1.306	1.417	1.269	1.468	1.233	1.483	1.066	1.831
16	1.327	1.436	1.309	1.495	1.283	1.515	1.103	1.886
17	1.351	1.491	1.344	1.572	1.326	1.595	1.427	1.987
18	1.380	1.517	1.376	1.600	1.354	1.629	1.431	2.039
19	1.417	1.554	1.427	1.641	1.413	1.672	1.475	2.079
20	1.445	1.574	1.466	1.667	1.456	1.697	1.686	2.142
125	8.04	5.46	10.36	6.51	10.81	6.83	18.34	10.41

TABLE VII
AVERAGE SPICE SUBOPTIMALITY OF ELMORE DELAY AS MEASURED BY THE RATIO BETWEEN THE AVERAGE SPICE DELAYS OF THE ELMORE-OPTIMAL AND SPICE-OPTIMAL TOPOLOGIES

	Critical Sink Delay		Maximum Sink Delay	
Technology	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	1.029	1.099	1.009	1.001
IC2	1.039	1.096	1.005	1.002
IC3	1.038	1.078	1.013	1.002
MCM	1.019	1.031	1.001	1.001

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 surprisingly high fidelity for the critical-sink delay criterion, and nearly perfect fidelity for the maximum sink delay criterion. For example, with 5-pin nets and IC3 technology parameters, optimal critical-sink topologies under Elmore delay averaged only 5.6 rank positions (out of 125) away from optimal according to SPICE, while the best topology for maximum Elmore delay averaged only 0.2 positions away from its "proper" rank using SPICE. Reference [24] has similarly established the fidelity of Elmore delay for circuit design: they plotted Elmore- versus SPICE-computed delays for a suite of 209 different place/route solutions of the same ripple-carry adder circuit, and also found a very high correlation between the two delay measures.

Table VI shows the average increase in SPICE delay from optimal for the 20 top-ranking topologies, i.e., the 20 lowest SPICE delays for $|N| = 5$. For IC2, the average distance of 6.4 rank positions for the optimal critical sink Elmore topology implies an expected difference of approximately 1.6% in actual SPICE-computed delay (i.e., halfway between the seventh and eighth best topologies); for IC3 a distance of 5.6 rank positions implies approximately 0.7% delay suboptimality; and for MCM a difference of 5.1 rank positions implies 0.4% delay suboptimality. For maximum sink delay, Table VI implies approximate suboptimality ranging between 0.6% for MCM and 2.4% for IC3.

A more direct measure of the suboptimality of Elmore delay is to compare SPICE delays of the Elmore-optimal and SPICE-optimal topologies. Table VII shows averages of this measure of suboptimality for both the critical sink and the maximum delay criteria. For critical sink delay and $|N| = 5$, the average SPICE suboptimality of the Elmore-optimal topology is between 3.1% for MCM and 9.9% for IC1. Moreover, the Elmore-optimal topologies are closer to SPICE-optimal for the IC3 and MCM technologies, which have lower resistance ratios. We believe that the estimates of Elmore suboptimality for critical-sink delay in Table VII are larger than those inferred above from Tables IV-VI, due to the convexity of the relationship between SPICE rank and average SPICE delay (Table VI). For maximum delay, Table VII indicates that minimizing Elmore delay very nearly minimizes SPICE delay, with suboptimality of between 0.1% and 0.2% for the optimal Elmore topology. Thus, while the *accuracy* of Elmore delay has many dependencies on technology and is particularly weak for critical-sink delay, we find that the *fidelity* of Elmore delay is strong for both the critical-sink and maximum sink delay criteria.

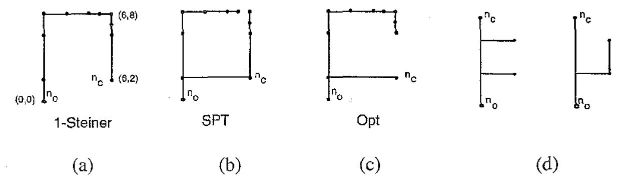


Fig. 1. (a)–(c) Optimal Steiner tree (cost 2.0 cm, $t(n_c) = 3.34$ ns); minimum cost shortest-paths tree (cost 2.5 cm, $t(n_c) = 2.26$ ns); and optimal-delay tree (cost 2.2 cm, $t(n_c) = 1.67$ ns) for the same sink set. Coordinates shown are in mm, and the 1.2μ IC2 technology parameters (Table I) were used with the two-pole simulator and 90% rise time delay criterion. (d) two distinct minimum-cost SPT solutions for a set of three sinks.

B. Intuitions from Elmore Delay

Because of its fidelity to SPICE-computed delay, Elmore delay is a good performance objective for constructing high-performance routing trees. Furthermore, the simplicity of the Elmore delay formula (1) allows us to intuit heuristics which effectively minimize delay.

Since r_{e_v} and c_{e_v} are usually proportional to the length of edge e_v, we see that $t_{\mathrm{ED}}(n_i)$ has a quadratic relationship to the length of the n_0-n_i path, suggesting a min-radius criterion. However, the C_j term implies that Elmore delay is also linear in the total edge length of the tree which lies outside the n_0-n_i path, suggesting a min-cost criterion. The relative size of the driver resistance r_d heavily influences the optimal routing topology: if r_d is large, the optimal routing tree (ORT) is a minimum cost tree; as r_d decreases, the ORT tends to resemble a "star" topology. The size of r_d relative to unit wire resistance is a "resistance ratio" [5] that captures the technology vis-a-vis routing tree design. Values of the resistance ratio are larger for current-generation CMOS, but tend to decrease in MCM substrate and some submicron CMOS IC interconnects (Table I).

In Fig. 1, we show a signal net N with identified critical sink n_c, along with three routing trees: (a) the 1-Steiner tree, (b) a minimum-cost SPT, and (c) the optimal CSRT with respect to critical sink n_c. Based on this example, the example of Fig. 1(d), and (1), we make the following observations:

- The minimum cost solution Fig. 1(a) has large delay to the critical sink n_c due to the long source-sink path.
- However, requiring a monotone path to *every* sink, as in the SPT Fig 1(b) or a Steiner arborescence [11], [30], can result in large tree capacitance which again leads to large delay at n_c.
- The optimal CSRT construction Fig. 1(c) illustrates the dependence of routing topology on the choice of critical sink, and reflects both the minimum-cost and the SPT solutions.
- Finally, (1) implies that the number of Steiner points in the n_0-n_c path should be minimized, and the Steiner points "shifted" toward n_0 (i.e., branches off of the n_0-n_c path should occur as close to the source as possible). Fig. 1(d) shows two trees which are both shortest-path trees and minimum Steiner trees, yet the rightmost tree has less signal delay at n_c.

```
CS-Steiner Algorithm
Input: signal net $N$; source $n_0 \in N$; identified critical sink $n_c \in N$
Output: heuristic CSRT solution $T$
1.  Construct heuristic minimum-cost tree $T_0$ over $N - n_c$.
2.  Form $T$ by adding a direct connection from $n_c$ to $T_0$,
      i.e., such that the $n_0$-$n_c$ path in $T$ is monotone.
```

Fig. 2. The CS-Steiner heuristic.

III. Two Classes of CSRT Heuristics

A. The CS-Steiner Approach

Given the observations above, we may characterize the optimal CSRT solution in Fig. 1(c) as one which minimizes total tree cost, *subject to the path from n_0 to n_c being monotone* (i.e., of minimum possible length). This simultaneous consideration of radius and cost parameters recalls the motivations in [1], [9], [10], but here the tradeoff is formulated with respect to the critical sink n_c. We thus obtain our *CS-Steiner* heuristic for the CSRT problem (Fig. 2).

The idea behind CS-Steiner is simple: construct a minimum-cost Steiner routing tree as usual, then "fix" the tree to reflect an identified critical sink. Since the algorithm template is quite general, we have examined a number of CS-Steiner variants. All of our variants use the 1-Steiner heuristic of [22] to construct the initial tree T_0 in Line 1. Section IV reports results for the following three variants:[8]

H0: The direct connection in Line 2 consists of a single wire from n_c to n_0.

H1: The direct connection in Line 2 consists of the shortest possible wire that can join n_c to T_0, subject to the monotone path constraint.

HBest: Accomplish Line 2 by trying all shortest connections from n_c to edges in T_0, as well as from n_c to n_0; perform timing analysis on each of these routing trees, and return the tree with lowest delay at n_c.

The time complexity of these variants is dominated by the construction of T_0 in Line 1 (or possibly by the simulator calls in HBest).

We enhance the CS-Steiner construction via an efficient *Global Slack Removal* (GSR) postprocessing algorithm. GSR [6] is similar to the method developed independently in [8], which also removes "U's" from interconnections. However, the objective of GSR is not to reduce tree cost (which is already effectively minimized by the 1-Steiner algorithm) but rather to maximize the monotonicity of all source-sink paths and reduce Elmore delay to all sinks. GSR accomplishes this

[8] We also studied two additional variants. **Variant H2** modifies Line 1 of CS-Steiner so that the initial heuristic tree T_0 is constructed over the entire net N. H2 then deletes the edge which lies directly above n_c when we root T_0 at n_0, and rejoins (the component containing) n_c to (the component containing) n_0 using a shortest possible wire from n_c, as in variant H1. **Variant H3** performs Lines 1 and 2 simultaneously by executing the 1-Steiner algorithm subject to a "maintaining monotone feasibility" constraint. In other words, we iteratively choose a Steiner point which minimizes the sum of the tree cost and the cost of any needed direct connection from n_c to n_0. The direct connection from n_c requires that there exist a monotone path through the "bounding boxes" of the edges in the path to n_0. Intuitively, this favors initial choice of Steiner nodes along some monotone path from n_0 and n_c, since such nodes will most rapidly reduce the marginal cost of adding the direct n_c-n_0 connection. The H2 and H3 variants yielded delays that were inferior to those of H0, H1, and HBest.

without increasing overall tree cost. For expository reasons, we defer formal description of GSR, along with its proofs of correctness, to Appendix A.

B. Elmore Routing Trees

From the discussion of Section II-B, we see that current routing objectives such as minimum tree cost, bounded tree radius, or prescribed cost-radius balance have often been *motivated* by the Elmore model. However, such objectives are abstractions: they do not directly optimize Elmore delay. Thus, the effectiveness of a given objective often depends on the prevailing technology, on the particular distribution of sink locations for a given signal net, and on the user's ability to find the parameter value (e.g., ϵ in the BRBC algorithm [10], or c in the AHHK algorithm [1]) which will yield a good solution for the particular input.

In this subsection, we depart from the abstraction inherent in "minimum cost" or "bounded radius" objectives, and propose a new greedy *Elmore routing tree* (ERT) approach which optimizes Elmore delay *directly* as the routing tree is constructed. The ERT approach is efficient, since Elmore delay at all nodes of a routing tree can be evaluated in linear time (see Footnote 2). Based on the performance results in Section IV for both critical-sink and "generic" performance-driven routing formulations, we believe that the ERT approach, which we have embodied as the ERT, SERT, and SERT-C algorithms described below, offers a basic new tool for VLSI routing.

The basic ERT approach is embodied in our Elmore routing tree (ERT) algorithm[9] for spanning trees (Fig. 3), which is analogous to Prim's minimum spanning tree construction [29]: 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 (n_i, n_j) yields a tree with minimum Elmore delay. The construction terminates when the entire net is spanned by the growing tree.[10] Note that greedy approach of the ERT algorithm can be generalized to any delay model by applying the appropriate estimator in Line 3 of Fig. 3.

We apply the ERT approach to Steiner routing by allowing the new pin to connect to an *edge* (or the source) of the existing tree, possibly inducing a Steiner node on this edge at the point that is closest to the new pin. In this way, the number of ways a pin outside the current tree can be added at each iteration is at most the number of edges in the current tree plus one

```
ERT Algorithm
Input: signal net $N$ with source $n_0 \in N$
Output: routing tree $T$ over $N$
1.  $T = (V, E) = (\{n_0\}, \emptyset)$
2.  While $|V| < |N|$ do
3.      Find $u \in V$ and $v \notin V$ which minimize the maximum Elmore delay
            from $n_0$ to any sink in the tree $(V \cup \{v\}, E \cup \{(u, v)\})$
4.      $V = V \cup \{v\}$
5.      $E = E \cup \{(u, v)\}$
6.  Output resulting spanning tree $T = (V, E)$
```

Fig. 3. The ERT Algorithm: Direct incorporation of the Elmore delay formula into a heuristic routing tree construction.

(i.e., a connection to the source). Note that the orientation of each "L-shaped" edge remains flexible until a Steiner node is placed on it.

- For *generic* performance-driven routing, our *Steiner Elmore routing tree* (SERT) algorithm iteratively finds $u \notin V$, $(v, v') \in E$, so that connecting u to the closest point on edge (v, v') minimizes the maximum source-sink Elmore delay in the resulting tree.
- To address *critical-sink* routing, our *Steiner Elmore routing tree with identified critical sink* (SERT-C) algorithm begins with a tree containing the single edge (n_0, n_c) in Line 1 of Fig. 3, then continues as in the SERT algorithm, except that we minimize $t_{\mathrm{ED}}(n_c)$ rather than the maximum delay to all sinks.

While CS-Steiner began with a minimum-cost Steiner tree and heuristically perturbed it to improve $t(n_c)$, SERT-C uses the opposite approach of starting with the required n_0-n_c connection and growing the routing tree while keeping $t_{\mathrm{ED}}(n_c)$ as small as possible. Again, we note that SERT-C offers a consistent, *direct* incorporation of Elmore delay within its construction, in contrast to heuristics whose objectives or strategies are only motivated by Elmore delay and whose solution quality may therefore be more sensitive to the technology, the input instance, and the choice of parameters.

Time complexities for our ERT variants are analyzed as follows.

Observation 1: The SERT-C algorithm can be implemented in $O(k^2 \log k)$ time.

Proof: The effect on delay $t_{\mathrm{ED}}(n_c)$ of inserting a new edge (u, w) into T arises only in the C_j terms in (1), and is an *additive constant* no matter when (u, w) is added into the tree. Initially, we compute the best connection from each noncritical sink to the tree containing only edge (n_0, n_c). For each new sink added, at most three new edges will be inserted into the tree. In constant time, we can calculate the effects of connections from a given sink outside T to these three new edges (all previously computed effects remain unchanged and need not be recomputed). We can insert the new delay effects into a priority queue for each $u \notin V$ in $O(\log k)$ time and also retrieve the current minimum-cost connection for v in $O(\log k)$ time. Thus, each pass through the **while** loop of Fig. 3 can be accomplished in $O(k \log k)$ time, giving an overall time complexity of $O(k^2 \log k)$. □

Observation 2: The ERT spanning tree algorithm can be implemented in $O(k^3)$ time, assuming constant unit wire resistance, unit wire capacitance, and sink capacitances.

Proof: The result follows from a simple observation: If a new tree edge incident to sink $u \in V$ (Line 3 of Fig. 3)

[9] Note that "ERT approach" refers to our basic concept of optimizing Elmore delay directly via a greedy heuristic. In contrast, the "ERT algorithm" is simply one of many possible implementations of the ERT approach: specifically, it is a greedy spanning tree construction.

[10] Our approach should be distinguished from the method of [28], wherein A^* heuristic search and the actual Elmore delay formula are used in a performance-driven routing tree construction. Like our method, [28] grows a routing tree over a net N starting from the source n_0; they perform A* search of a routing graph (e.g., in building-block design) to find the Elmore delay-optimal Steiner connection from the existing tree to a new sink. However, *the choice of this new sink is forced:* the algorithm always adds the sink that is closest (by Manhattan distance) to the existing tree, and thus falls into the standard pitfall of ignoring the underlying delay criterion. The effect of this difference is apparent in the ERT ordering of added nodes in Fig. 4. Indeed, the method of [28] can yield Elmore delays substantially larger than those of ERT: given a very tall, "hairpin"-like version of Fig. 1(a) with many sinks very closely spaced along the entire hairpin path, [28] forces the sinks to be added into the tree according to the path order (starting from the source n_0 at the lower left), yielding an obviously poor solution.

TABLE VIII
TWO-POLE SIMULATION RESULTS COMPARING CS-STEINER TREES WITH 1-STEINER HEURISTIC TREES. EACH ENTRY CORRESPONDS TO AN AVERAGE OVER DELAY COMPUTATIONS FOR RANDOM CRITICAL SINKS IN EACH OF 100 DIFFERENT RANDOM SIGNAL NETS. 1-STEINER RESULTS ARE REPORTED IN THE PHYSICAL UNITS (ns OR cm) WHILE OTHER RESULTS ARE REPORTED BY THEIR PERCENT DIFFERENCE FROM THE 1-STEINER RESULTS. NOTE THAT 1-STEINER AND 1-STEINER + GSR ALWAYS PRODUCED NEARLY IDENTICAL AVERAGE COSTS

		IC1		IC2		IC3		MCM	
		$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$
Critical Sink Delay	1Stein	0.549 ns	0.848 ns	0.331 ns	0.520 ns	0.218 ns	0.342 ns	2.31 ns	4.09 ns
	1Stein+GSR	-2.2%	-3.6%	-3.0%	-6.6%	-3.2%	-5.0%	-4.8%	-7.3%
	H0+GSR	-2.0%	-17.6%	-12.4%	-30.0%	-15.1%	-33.6%	-45.0%	-66.7%
	H1+GSR	-4.0%	-11.7%	-6.6%	-17.3%	-7.8%	-19.0%	-14.3%	-33.5%
	HBest+GSR	-7.1%	-18.3%	-13.3%	-27.9%	-15.6%	-30.7%	-40.7%	-66.0%
Ave WL	1Stein	1.48 cm	2.18 cm	1.48 cm	2.18 cm	1.48 cm	2.18 cm	14.8 cm	21.8 cm
	H0+GSR	+29%	+22%	+29%	+22%	+29%	+22%	+29%	+22%
	H1+GSR	+4%	+6%	+4%	+6%	+4%	+6%	+4%	+6%
	HBest+GSR	+7%	+11%	+10%	+12%	11%	+12%	+22%	+21%

minimizes the maximum Elmore delay $\max_i t_{\mathrm{ED}}(n_i)$, it must connect u to the sink $v \notin V$ that is closest to u. Thus, at each pass through the **while** loop, we simply compute the shortest "outside connection" for each node in V, i.e., every possible u, in $O(k^2)$ time. We then add each of the $O(k)$ shortest outside connections to T in turn. Evaluating the Elmore delays at all sinks in each of the resulting trees requires $O(k)$ time per tree. Hence, each pass through the **while** loop requires $O(k^2)$ time, and this yields the $O(k^3)$ complexity result.[11] □

In practice, the complexity of the ERT algorithm will be transparent to the user, since k is typically small (e.g., our runtimes for the problem sizes discussed here are 0.01 s on Sun SPARC1 hardware; see also Footnote 18). We know of no implementation of the SERT algorithm that is faster than $O(k^4)$. Intuitively, the difficulty is that 1) in Line 3 we must always consider $\Theta(k^2)$ Steiner connections, and 2) the connection which minimizes $\max_i t_{\mathrm{ED}}(n_i)$ in Line 3 may not be the best one from the "perspective" of any individual sink in N or edge in T. Thus, we currently have a rather interesting situation where the CSRT problem formulation leads to an algorithm (SERT-C) that enjoys nearly quadratic speedup over the generic Steiner computation (SERT).

IV. EXPERIMENTAL RESULTS

A. CS-Steiner Trees

We implemented each of the CS-Steiner variants H0, H1 and HBest, along with the 1-Steiner algorithm [22], using C on a Sun SPARC1 ELC workstation, and ran these algorithms on random 4- and 8-sink inputs.[12] We also applied our GSR post-processing algorithm (denoted as +GSR) to 1-Steiner and each of the CS-Steiner variants. Our inputs correspond to the four distinct technologies described in Table I.

Table VIII gives delay and tree cost (WL) results and comparisons. The delays at all sink nodes correspond to 50% rise times estimated using the Two-Pole simulator [37], [38]. Each entry in Table VIII represents an average taken over every sink node in 50 random point sets. We emphasize that the 1-Steiner algorithm (or the BRBC, AHHK, etc. methods), being net-oriented, will return the same tree for a given sink set no matter which sink happens to be critical; the delays at the sinks n_i are in some sense "generic." In contrast, each of the three CS-Steiner variants can return a different tree for each choice of critical sink in the same net. Thus, for each variant we report the delay at n_i in the *specific* tree corresponding to identification of n_i as the critical sink.

Variants H0 and HBest significantly reduce delay to the critical sink, particularly in larger nets and for MCM interconnect technology where output driver and wire resistances are low. In other words, the simple strategy of connecting the critical node via a path with low branching factor is very successful for these cases. Of course, this strategy will produce larger routing cost.[13]

B. Elmore Routing Trees

We constructed Elmore routing trees for the same sets of random inputs used in the CS-Steiner experiments. Delay simulation results, again obtained using the Two-Pole simulator, are presented in the upper parts of Table IX. For comparison, the table includes data for the minimum spanning tree and AHHK tree [1] constructions.

Our results show that even as generic net-dependent routers, the ERT methods we propose are highly effective, beyond their relative efficiency and ease of implementation. For nets with nine sinks, the spanning tree ERT construction reduces critical sink delay versus the MST construction by 16%, 26%, and 30% in the respective IC technologies and by 67% in the MCM technology. ERT also improves upon AHHK for most of the technologies, with reductions of 0% (IC1), 4% (IC2), 6% (IC3), and 46% (MCM). These results are particularly impressive because our AHHK data follows the experimental methodology in [1], which generates output trees for 21 different values of the c parameter and then chooses the best tree found for each signal net instance.[14]

[11] Again, we note the fundamental difference between the ERT approach and the method of [28]: while [28] must add the single sink that is closest to the existing tree, the ERT algorithm identifies both a new sink and its connection such that Elmore delay is minimized.

[12] Results for 16-sink inputs have been reported in preliminary form, e.g., [6]. While such large inputs magnify the effect of our new methods, in practice most signal nets will be within the size range that we now discuss.

[13] Highly "star-like" topologies can possibly introduce other difficulties such as crossing wires, nodes with degree > 4, and capacitive coupling effects; these are not modeled by either SPICE or the Two-Pole simulator. Note that HBest uses calls to the Two-Pole simulator in its delay analysis for candidate connections; see the definition of HBest in Section III-A.

[14] According to [1], AHHK already achieves strong improvements over such other recent methods as shallow-light routing [10] or Steiner arborescences [11] when measured by the same Two-Pole simulation methodology. However, it should also be noted that delay reductions in practice will probably not attain exactly these magnitudes, partly because our modeling methodology cannot capture all of the device characteristics and delay effects related to the geometric embedding of our topologies.

TABLE IX
TWO-POLE SIMULATION RESULTS FOR ELMORE ROUTING TREE VARIANTS. SPANNING ERT CONSTRUCTIONS ARE COMPARED WITH MST AND AHHK; STEINER SERT AND SERT-C CONSTRUCTIONS ARE COMPARED WITH 1-STEINER. ALL CHOICES OF CRITICAL SINK ARE RANDOM, AND ALL RESULTS ARE AVERAGED OVER 100 RANDOM NETS. MST AND 1-STEINER RESULTS ARE REPORTED IN THE PHYSICAL UNITS (ns OR cm) WHILE OTHER RESULTS ARE REPORTED AS PERCENT DIFFERENCES FROM CORRESPONDING MST OR 1-STEINER RESULTS

		IC1		IC2	
		$\|N\| = 5$	$\|N\| = 9$	$\|N\| = 5$	$\|N\| = 9$
Crit. Sink Delay	MST	0.645 ns	0.984 ns	0.395 ns	0.609 ns
	AHHK	-9.6%	-16.3%	-13.7%	-23.0%
	ERT	-12.1%	-16.3%	-19.6%	-25.9%
	1Stein	0.549 ns	0.848 ns	0.331 ns	0.520 ns
	SERT	-3.3%	-11.6%	-7.9%	-19.4%
	SERT-C	-5.3%	-15.3%	-13.0%	-26.5%
Max Delay	MST	0.758 ns	1.213 ns	0.485 ns	0.792 ns
	AHHK	-12.4%	-19.5%	-16.5%	-25.3%
	ERT	-14.5%	-21.0%	-21.4%	-30.1%
	1Stein	0.627 ns	1.028 ns	0.393 ns	0.664 ns
	SERT	-4.5%	-14.7%	-8.1%	-22.0%
	SERT-C	-3.0%	-8.6%	-3.8%	-10.8%
Ave WL	MST	1.64 cm	2.43 cm	1.64 cm	2.43 cm
	AHHK	+16%	+9%	+16%	+9%
	ERT	+10%	+15%	+18%	+25%
	1Stein	1.48 cm	2.18 cm	1.48 cm	2.18 cm
	SERT	+6%	+9%	+11%	+18%
	SERT-C	+6%	+6%	+15%	+11%

		IC3		MCM	
		$\|N\| = 5$	$\|N\| = 9$	$\|N\| = 5$	$\|N\| = 9$
Crit. Sink Delay	MST	0.262 ns	0.403 ns	2.82 ns	4.80 ns
	AHHK	-11.5%	-25.1%	-22.3%	-39.2%
	ERT	-21.8%	-29.8%	-52.8%	-67.1%
	1Stein	0.218 ns	0.342 ns	2.31 ns	4.09 ns
	SERT	-9.2%	-21.9%	-41.6%	-61.6%
	SERT-C	-16.1%	-30.7%	-43.3%	-66.0%
Max Delay	MST	0.326 ns	0.533 ns	3.86 ns	7.05 ns
	AHHK	-17.8%	-27.0%	-24.1%	-36.8%
	ERT	-23.6%	-33.2%	-45.6%	-60.1%
	1Stein	0.262 ns	0.444 ns	3.06 ns	5.92 ns
	SERT	-9.2%	-24.1%	-30.1%	-51.9%
	SERT-C	-4.6%	-10.8%	-14.1%	-15.4%
Ave WL	MST	1.64 cm	2.43 cm	16.4 cm	24.3 cm
	AHHK	+16%	+9%	+4%	1.07
	ERT	+19%	+27%	+61%	2.15
	1Stein	1.48 cm	2.18 cm	14.8 cm	21.8 cm
	SERT	+13%	+22%	+66%	+127%
	SERT-C	+16%	+14%	+28%	+22%

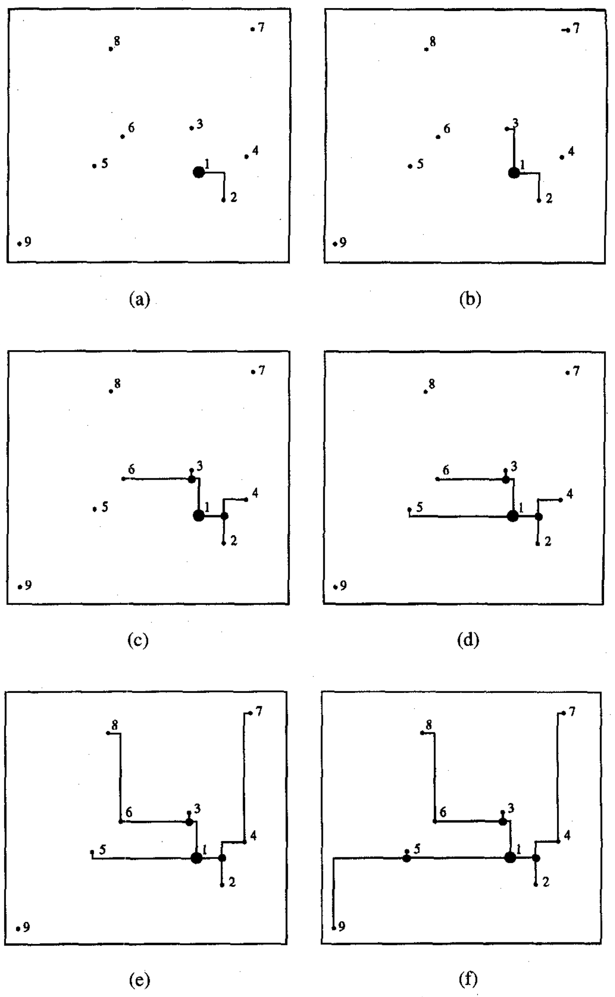


Fig. 4. Example of the progressive SERT Steiner tree construction for a 9-terminal net using IC2 parameters. The source pin is labeled 1, and sinks are numbered in order of distance from the source.

The Steiner ERT variant also performs well as a generic high-performance router. For 9-pin nets, SERT improves critical sink delay versus the 1-Steiner routing by 19% and 62% for the IC2 and MCM technologies, respectively. The percentage reductions in maximum delay are somewhat greater for the IC technologies, but somewhat smaller for MCM interconnects. It should be noted that for the MCM technology, the ERT and SERT constructions tend to be star-like, producing tree costs significantly higher than those of the 1-Steiner construction. In practice, when delay is not an overriding concern, the user may recapture wirelength by simulating a larger output driver resistance.

Finally, even more significant reductions in delay can be achieved when a critical sink has been identified per the original CSRT formulation. The SERT-C algorithm improves over the SERT results by an *additional* reduction in delay at the critical sink of 5%, 7%, and 6% for the three IC technologies, and 8% for MCM. Identification of a critical sink has clear advantages in terms of tree cost, particularly for MCM routing: the SERT-C trees have much less cost than the SERT outputs, while still improving the delay to the critical sink. Since maximum sink delays still decrease, it is likely that overall skew in the routing tree will be reduced even when we treat the critical-sink formulation. Finally, we note that the SERT-C router produces very similar delays and costs compared to the HBest and H0 variants of CS-Steiner discussed in the previous subsection. However, SERT-C is more practical than HBest or H0 since it runs in $O(k^2 \log k)$ time (versus the $O(k^3)$ complexity of the best practical implementation of the 1-Steiner heuristic that is called by HBest and H0), and it does not require any simulator calls as does HBest.

Figs. 4 and 5 illustrate the SERT and SERT-C algorithms for a 9-pin signal net using the IC2 technology parameters. Fig. 4 shows the progressive growth of the SERT construction. Fig. 5 contains the trees produced by SERT-C for the various choices of critical node. The tree constructed when n_c is node

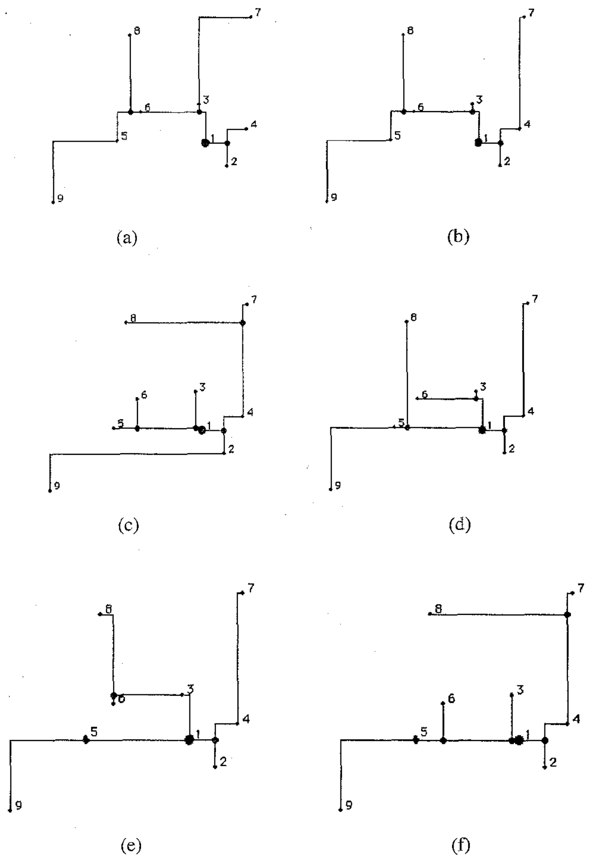


Fig. 5. SERT-C tree constructions for a single 9-pin net, showing variation of solution with choice of critical sink n_c. (a) Node 2 (or 4) critical. (b) Node 3 (or 7) critical (also 1–Steiner tree). (c) Node 5 critical. (d) Node 6 critical. (e) Node 8 critical (also Steiner ERT). (f) Node 9 critical.

3 or node 7 is also the 1-Steiner tree, and the tree constructed when n_c is node 8 is the same as the generic SERT result.

C. Elmore-Optimality of Spanning Tree Constructions

We have seen that the ERT constructions yield greatly improved signal delay when compared to previous methods. An obvious question is whether we still need to seek methods that better minimize Elmore delay. Thus, we have implemented a branch-and-bound algorithm which finds *optimal* generic routing trees according to Elmore delay. Starting with a trivial tree containing only the source pin, we incrementally add one edge at a time to the growing tree and evaluate the maximum sink delay. If this value exceeds the maximum sink delay in any *complete* candidate tree seen so far, we prune the search and backtrack to select a different edge at the previous step. A recursive implementation of this *Branch-and-Bound Optimal Routing Tree* (BBORT) search is shown in Fig. 6. BBORT attempts to add sinks in all possible orders, but avoids testing any topology more than once by requiring that sinks be added in the order of a breadth-first traversal of the tree (if two sinks are connected to the same parent node, then the sink with smaller index must be added to the tree first). It is easy to verify that according to this convention, any tree topology will imply a unique ordering of the sinks. Consequently, although BBORT tries all possible orderings of sinks, it calculates delay

```
BBORT Algorithm
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 $t(T') < t_{\min}$ Then
8.         If $|T'| = |N|$ Then $T_{opt} = T'$ ; $t_{\min} = t(T')$
9.         Else Call Add_Edges($T'$)
```

Fig. 6. Branch-and-Bound Optimal Routing Tree (BBORT) algorithm (recursive implementation).

TABLE X
ELMORE DELAYS AND WIRELENGTHS OF VARIOUS CONSTRUCTIONS USING IC1, IC2, IC3, AND MCM PARAMETERS. SIMULATIONS WERE RUN ON 200 RANDOM NETS FOR EACH NET SIZE. TREE COST IS NORMALIZED TO MST COST AND DELAYS ARE NORMALIZED TO ORT DELAY. STANDARD ERRORS FOR ERT DELAY ARE SHOWN IN PARENTHESES

	IC1				IC2			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
ORT	1.0	1.103	1.0	1.133	1.0	1.140	1.0	1.175
ERT	1.007	1.104	1.017	1.142	1.010	1.159	1.022	1.215
(Std Err)	(.0015)		(.0021)		(.0017)		(.0022)	
SPT	1.085	1.290	1.130	1.395	1.058	1.290	1.096	1.395
MST	1.169	1.0	1.282	1.0	1.272	1.0	1.451	1.0

	IC3				MCM			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
ORT	1.0	1.146	1.0	1.190	1.0	1.432	1.0	1.547
ERT	1.011	1.172	1.027	1.252	1.009	1.585	1.024	1.892
(Std Err)	(.0018)		(.0025)		(.001)		(.0008)	
SPT	1.054	1.290	1.091	1.395	1.089	1.290	1.161	1.395
MST	1.311	1.0	1.499	1.0	1.894	1.0	2.457	1.0

at most once for each tree topology. In Fig. 6, lines 7 through 9 comprise the core of the branch-and-bound methodology used: if the delay in the current tree T' is greater than or equal to $t_{\min}$ (the current best-known delay for a complete tree), then procedure Add_Edges terminates and the algorithm backtracks. Otherwise, if T' is a complete tree, then $t_{\min}$ is set to the delay of T', or if T' is a partial tree, then Add_Edges recursively adds more edges to T'.

To track all of the above simulation results, we have run BBORT trials on random sets of 200 nets for each of several net sizes. Our inputs are evaluated using the same four sets of technology parameters discussed previously. Table X compares Elmore delays of the BBORT and ERT constructions, as well as of the minimum spanning tree (MST) and shortest path tree (SPT) constructions, for each of the four technologies.[15] Delay for each tree is normalized to the ORT delay of the same net. Tree costs are similarly normalized to the MST cost of each net.

In Table X, we see that ERT's over seven pins in the IC1 technology have an average maximum Elmore delay only 1.7% greater than optimal, while MST's have average Elmore delay 28.2% greater than optimal. For smaller nets, ERT's are even better: for nets with five pins, ERT delays are only 0.7% above optimal on average, while MST's are 16.9% above

[15]The SPT construction is the tree which minimizes cost subject to each source/sink path having minimum length, i.e., it is a Steiner arborescence, or A-tree [11], [30].

optimal. Our confidence in the average difference computed between ERT's and ORT's is very high. For instance, the 1.7% difference obtained for 7 pins has a standard error[16] of 0.21%, indicating a 95% confidence interval between 1.3% and 2.1% (i.e., an interval of within two times the standard error of the average).

Technology IC3 gives our worst results in terms of the optimality of ERT's. For the IC3 parameters and 7-pin nets, ERT gives an average value within 2.7% of ORT with a 95% confidence interval of between 2.2% and 3.2%. For MCM parameters, the Elmore-based ERT constructions are also very close to optimal: on average, they are within 2.4% of ORT delay for 7-pin nets. Finally, our tables compare the delays of the SPT construction with those of the ERT and MST solutions; the SPT outperforms the MST, but not the ERT, in terms of Elmore delay.

D. Elmore-Optimality of Steiner Tree Constructions

We have shown that our spanning tree constructions are nearly optimal when we optimize the maximum Elmore delay over all sinks in the net. Because Steiner constructions give lower delay values than spanning trees in general, we close this section with a similar comparison for our SERT-C and SERT Steiner constructions. At first, this comparison appears very complicated because there are infinitely many possible locations for Steiner nodes. Indeed, while it is well-known that the result of [17] restricts the choice of Steiner nodes in a minimum-cost Steiner tree to at most $k \cdot (k+1)$ points, no such characterization has been established for a Steiner tree with optimal *Elmore* delay. In Appendix B, we present new theoretical results which restrict the choice of Steiner nodes in Elmore-optimal trees to exactly the same finite "Hanan grid" that contains the Steiner nodes of minimum-cost trees. This allows a finite algorithm which determines *optimal* trees with respect to any given linear combination of Elmore delays to critical sinks. We also present an entirely new "peeling decomposition" of any optimal Elmore delay Steiner tree into a sequence of subtrees, each of which adds a sink by a "closest connection" to some edge in the previous tree.

When the driver resistance r_d is very large, the optimal Elmore delay tree is a minimum-cost Steiner tree (1), [5]. As a consequence, our results extend very naturally to the well-studied problem of minimum-cost Steiner tree construction, and the restriction of Elmore-optimal Steiner nodes to the Hanan grid both generalizes and extends Hanan's original results. (As Hanan did for minimum-cost Steiner trees, we prove that every Steiner node in an Elmore-optimal tree is connected to one sink by a horizontal segment of edges, and to another sink by a vertical segment of edges. However, our techniques (Lemmas B1–B4 in Appendix B) are much more powerful in order to address the optimality of the Steiner tree with respect to Elmore delay.) Our peeling decomposition and its extension to minimum-cost Steiner trees are of independent interest since they provide both a new characterization of, and a new means of generating, such trees.

TABLE XI
EMPIRICAL STUDY OF AVERAGE SUBOPTIMALITY OF THE SERT-C HEURISTIC IN TERMS OF ELMORE DELAY, USING IC1, IC2, IC3, AND MCM PARAMETERS. SIMULATIONS WERE RUN ON 200 RANDOM NETS FOR EACH NET SIZE. DELAY IS NORMALIZED TO BB-SORT-C DELAY AND TREE COST IS NORMALIZED TO 1-STEINER COST. STANDARD ERRORS FOR AVERAGE SERT-C DELAYS ARE SHOWN IN PARENTHESES

	IC1				IC2			
	$\lvert N\rvert = 5$		$\lvert N\rvert = 7$		$\lvert N\rvert = 5$		$\lvert N\rvert = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
SORT-C	1.0	1.111	1.0	1.112	1.0	1.161	1.0	1.158
SERT-C	1.042	1.046	1.083	1.047	1.049	1.120	1.114	1.106
(Std Err)	(.004)		(.006)		(.006)		(.009)	
1-Steiner	1.117	1.0	1.200	1.0	1.228	1.0	1.362	1.0
	IC3				MCM			
	$\lvert N\rvert = 5$		$\lvert N\rvert = 7$		$\lvert N\rvert = 5$		$\lvert N\rvert = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
SORT-C	1.0	1.175	1.0	1.165	1.0	1.296	1.0	1.262
SERT-C	1.046	1.140	1.112	1.112	1.000	1.296	1.001	1.256
(Std Err)	(.006)		(.010)		(.000)		(.0001)	
1-Steiner	1.275	1.0	1.429	1.0	1.455	1.0	1.634	1.0

Based on the results of Appendix B, we achieve a simple modification to our BBORT method which finds an optimal Steiner routing tree for any linear combination of Elmore delays to critical sinks. Rather than considering connections from each sink n_j outside the current tree to each sink n_i inside the tree as in BBORT, the *branch-and-bound* method for *Steiner optimal routing trees with critical sinks* (BB-SORT-C) considers connections from n_j to each edge created when n_i was added to the tree. In other words, each node n_i already contained in T is replaced as a possible connection point by each of the edges created when n_i was added to the tree earlier. Again we use branch-and-bound pruning to reduce the complexity of the search.[17]

Table XI compares Elmore delay for trees constructed by the SERT-C algorithm with optimal Elmore delay trees found by BB-SORT-C for each of the four technologies. The size of nets used in the comparison is limited to nets with six sinks (i.e., seven pins) because of the exponential time complexity of BB-SORT-C. For nets with seven pins, our results show that SERT-C achieves Elmore delay that is on average within 11.1% of optimal for the IC1 technology; results for IC2, IC3, and MCM parameters are very similar. The table also gives average tree costs for our constructions and the standard error of our estimate for the ratio between SERT-C and SORT-C delays.[18] We see that the SERT-C algorithm does not perform as well as the ERT algorithm in terms of nearness to optimality for the types of delay measures we have considered. Nevertheless, our results provide strong guidance for future efforts in performance-driven routing: even if future work improves the near-optimality of critical

[16]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 ERT and ORT delays is equivalent to the standard error of average ERT delay.

[17]Because we consider connections to up to three edges for each sink in the growing tree, our BB-SORT-C will introduce some redundancies in the tree topologies; we check for possible redundancies and prune the search at each redundant tree we find.

[18]Average running times for each 5-pin net (in seconds) are 0.006 (BB-SORT-C), 0.0004 (SERT-C), 0.0014 (SERT), and 0.0012 (1-Steiner). Average running times for 7-pin nets are 0.44 (BB-SORT-C), 0.0007 (SERT-C), 0.0055 (SERT), and 0.0035 (1-Steiner).

sink routing constructions, Table XI shows that any future improvement in Elmore delay will be at most from 8% to 12% for nets with up to seven pins.

V. Conclusion

We have addressed a *critical-sink routing tree* (CSRT) formulation which arises when critical-path information becomes available during the timing-driven layout process. Two new classes of CSRT constructions are proposed: 1) the CS-Steiner method, which modifies a minimum Steiner tree to accommodate an identified critical sink, and 2) the SERT-C method, which begins with a connection from the source to the critical sink and then grows a tree so as to minimize the increase in Elmore delay to the critical sink. Each of these algorithms is efficient, and offers very significant performance improvements over existing performance-driven routing tree constructions. We note that the greedy "Elmore routing tree" (ERT) approach underlying the SERT-C algorithm seems quite powerful. In particular, the approach encompasses a "generic" SERT Steiner router which outperforms all previous performance-driven routing algorithms in the literature. The ERT approach is also the first to *consistently*, and directly, optimize the Elmore delay formula itself, rather than an objective which heuristically abstracts Elmore delay. Since Elmore routing trees are efficiently computed, our approaches may lead to basic new utilities that can be integrated within existing performance-driven global routing codes. Assessments of the near-optimality of our Steiner constructions have led to a new characterization of Elmore-optimal Steiner trees, and to a new decomposition theorem for minimum-cost and minimum-Elmore delay Steiner trees; both of these results are of independent interest.

Which of our routing heuristics is most useful will depend on the application. The CS-Steiner heuristics H0 and HBest yield the smallest delay values for a single critical sink, but have high time complexity which may make them impractical for repeated wiring of large nets. Our SERT-C heuristic has time complexity of only $\Theta(k^2 \log k)$ and is readily extended to the case of nets with multiple critical sinks (e.g., first apply SERT with its min-max delay objective to the critical sinks, then apply SERT-C with a weighted average delay objective to connect the remaining sinks). The SERT heuristic can also be applied before critical path information becomes available (an alternative is the ERT spanning tree heuristic, which has lower time complexity but does not introduce Steiner nodes). For nets on noncritical paths, minimizing wire length can take precedence over minimizing delay, hence traditional minimum-cost Steiner heuristics such as 1-Steiner [22] are likely to be preferred.

Our heuristics which optimize Elmore delay directly are near-optimal in terms of SPICE-computed delay: we show that our methods give Elmore delay that is nearly optimal, and we also showed that Elmore delay-optimal trees have nearly optimal SPICE delay. For spanning trees with five pins, we estimate that the optimal tree according to Elmore delay will be between 3% and 10% above SPICE-optimal, depending on the technology. Given that our SERT-C heuristic is between

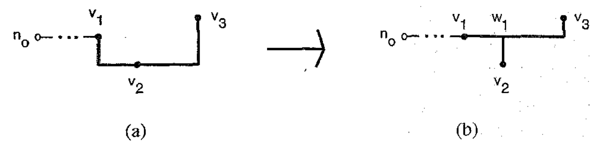


Fig. 7. Removing a single "V" in the GSR algorithm.

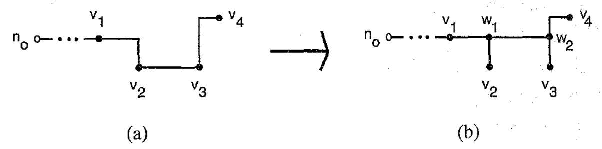


Fig. 8. Removing a single "U" in the GSR algorithm.

0% and 5% above optimal in terms of Elmore delay for five-pin nets, we estimate that the SPICE delay suboptimality of our SERT-C heuristic ranges from 3% for MCM to about 12% for 0.5 μm and about 15% for 1.2 and 2.0 μm CMOS IC technologies.

Current work addresses interesting extensions of the CS-Steiner and ERT approaches to incorporate wiresizing, address general-cell layout with arbitrary routing region costs, and exploit the inherent parallelizability of our approaches. Similar approaches may also apply to clock routing, although the extension is nontrivial because of larger net sizes and the addition of a minimum skew objective. Finally, we leave as an open problem the reduction in time complexity of the ERT constructions.

Appendix A
Global Slack Removal

Recall from Section III–A that *global slack removal* (GSR) is an efficient post-processing enhancement to the CS-Steiner approach. The worst-case complexity of GSR is $O(k^2)$, although we believe the average-case complexity to be very close to $O(k)$. GSR is a linear-time postprocessing enhancement to the CS-Steiner approach. GSR shifts edges in the 1-Steiner output to maximize the monotonicity of all source-sink paths without any increase in total tree cost or Elmore delay to any sink. In what follows, we use the term *1-Steiner tree* to refer to any tree that can be output by the 1-Steiner algorithm.

Definition: A V is a subpath of three consecutive nodes on a root-leaf path in a routing tree such that the combined edge cost along the subpath is greater than the distance between its two end points [e.g., path v_1–v_3 in Fig. 7(a)].

Definition: A U is a subpath of four consecutive nodes on a root-leaf path with edge cost greater than the distance between its end points [e.g., path v_1–v_4 in Fig. 8(a)].

Note that the nodes in a V or U can be either Steiner nodes or pins. A V can be removed by introducing a Steiner node which eliminates the overlap between the two adjacent edges, as in Fig. 7(b). It is easy to see that, if a U (say $v_1v_2v_3v_4$) does not contain any V's, then its middle edge (v_2, v_3) must be either completely horizontal or vertical. Consequently, a U containing no V's can be removed by moving the middle edge and adding up to two new Steiner nodes as in Fig. 8(b).

```
GSR Algorithm
Input: Steiner tree $T$ with source $n_0$
Output: Steiner tree $T$ with all $U$'s removed
1.   Remove all Steiner nodes of degree $\leq 2$ from $T$;
2.   $Q \leftarrow \{n_0\}$;
3.   While $Q \neq \emptyset$
4.       $v \leftarrow Dequeue(Q)$;
5.       For each node $v' \in children(v)$ do
6.           $Q \leftarrow Enqueue(v')$;
7.           If there is a $V$ located at $v'$
8.               Call Remove_V($v'$)
9.           If there is a $U$ located at $v'$
10.              Call Remove_U($v'$)
11.              Call Clean_Up($v'$)
12.      Remove all Steiner nodes of degree $\leq 2$ from $T$;

Subroutine Clean_Up(node: $v'$)
C1.  If there is a $V$ located at $parent(v')$
C2.      Call Remove_V($parent(v')$)
C3.  If there is a $U$ located at $v'$
C4.      Call Remove_U($v'$)
C5.      Call Clean_Up($v'$)
C6.  Else
C7.      If there is a $U$ located at $parent(v')$
C8.          Call Remove_U($parent(v')$)
C9.          Call Clean_Up($parent(v')$)
```

Fig. 9. Pseudo-code for the global slack removal (GSR) algorithm. Local variables include a queue Q and nodes v and v'. We use *children*(v) to denote the set nodes that are children of v when the tree is rooted at n_0; *parent* (v) denotes the parent of v in the rooted tree. The subroutine **Remove_V**(v) removes a V located at v as in Fig. 7 and **Remove_U** (v) removes a U located at v as in Fig. 8.

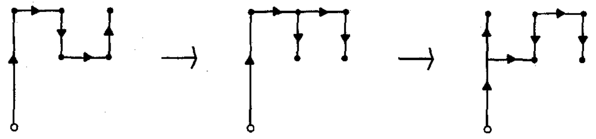


Fig. 10. An example of a net with a source and five sinks for which processing the U's in a bottom-up order returns a tree with one remaining U.

Fig. 9 describes the GSR algorithm for removing V's and U's from any Steiner tree. We define a U (or V) to be *located at* node v if v is the node in the U (or V) furthest topologically from the source. Three clarifying points should be noted. 1) GSR uses a "queue" Q which can be implemented arbitrarily as long as each node in the tree is processed before its children. In practice, a simple depth-first ordering suffices. 2) The procedure **Remove_U** is invoked only for U's not containing any V's, and is executed as in Fig. 8. 3) All *low-degree* Steiner nodes of degree ≤ 2 are clearly superfluous and are removed, since more U's can be found if they are deleted at the outset. Because U removal can introduce additional low-degree Steiner nodes, they are again removed at the end of the algorithm.

We now show that the tree returned by GSR dominates the input tree in terms of total tree cost, path length from the source to each sink, and Elmore delay at each sink. Let cost (T) denote the cost of routing tree T.

Theorem A1: Given any tree T as input, GSR will return a tree T' such that (i) cost$(T') \leq$ cost(T); (ii) for each $i > 0$, the n_0-n_i path length in T' is less than or equal to the n_0-n_i path length in T; and (iii) the Elmore delay $t_{\mathrm{ED}}(n_i)$ at each n_i in T' is less than or equal to the Elmore delay $t_{\mathrm{ED}}(n_i)$ in T.

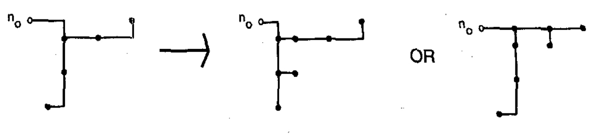


Fig. 11. The GSR algorithm with input (a) can produce either tree (b) or tree (c), depending on the order in which the U's are processed.

Proof: 1) Removing a V reduces cost in the routing tree; removing a U as in Fig. 8 leaves tree cost unchanged; and by the triangle inequality the removal of a low-degree Steiner point will either reduce cost or leave it unchanged. These are the only operations on the tree by GSR.

2) **Remove_V** reduces the source-sink path length to v_3 in the V and to all of its descendents; similarly, **Remove_U** reduces the source-sink path length to node v_4 in the U. Other source-sink path lengths remain unchanged in either procedure. Removing low-degree Steiner nodes does not affect any source-sink path lengths.

3) Assuming constant technology parameters,[19] removing a U or a V can affect Elmore delay along a source-sink path in only three ways: a) changing the length of the path; b) changing tree capacitances along the path (i.e., increasing the wirelength of branches off from the path); and c) shifting tree capacitances along the path (changing where branches connect to the path). Removing a V will reduce some path lengths, reduce tree capacitances, and shift tree capacitances closer to the source, thereby reducing Elmore delay to all pins in the tree. Removing a U reduces path length to node v_4 in Fig. 8 and shifts tree capacitance closer to the source for nodes v_2, v_3, and v_4. (For v_3, the capacitance that met the n_0-v_3 path at v_3 now meets the path at w_1 and w_2.) The only possible effect of removing low-degree Steiner nodes is to reduce total wirelength, which cannot increase delay to any sink. □

The order in which U's are removed from the tree is important. If the U's were processed in a bottom-up rather than a top-down order, then new U's could be introduced and the output tree might still contain U's, as in Fig. 10.[20] Note also that two different top-down orderings can produce different outputs (although neither will contain any U's; see Fig. 11).

We now prove that GSR removes all V's and all U's from any input tree, and that its worst-case time complexity is quadratic. Note that we have constructed a class of nets for which the 1-Steiner heuristic constructs a tree which GSR processes in $\Omega(k^2)$ time [6]. GSR in practice, however, seems to exhibit close to linear-time complexity, because multiple calls to procedure **Clean_Up** occur for very few nodes.

Theorem A2 1) GSR returns a tree containing no V's and no U's, and 2) GSR runs in $O(n^2)$ time in the worst case.

Proof: Since GSR checks for V's and U's at each node in the tree, the output tree will contain a V or U only if GSR creates one at a node that has already been traversed. A new V or U can be produced at a node v only if the n_0-v path

[19]I.e., including unit wire resistance, unit wire capacitance, driver resistance, and sink loading capacitances.

[20]By "bottom-up" we mean that each node is processed *after* all of its children in the tree, while a "top-down" ordering implies that each node is traversed *before* any of its children.

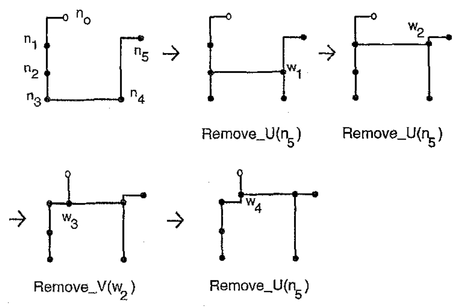


Fig. 12. Example in which removing a U at n_5 requires two subsequent U-removals and a V-removal to complete the **Clean_Up** procedure.

length is increased (which is impossible by Theorem 1) or if nodes are removed from the n_0-v path.

Removing a V at Line 8 in Fig. 9 will not introduce a new V or U at v_2 (in Fig. 7), because the n_0-v_2 path length is unchanged and a new Steiner point w_1 is added to this path. Removing a V will not introduce a V at v_3 either, because $v_1w_1v_3$ is not a V. A U may remain at v_3 after removing the V, but this will be detected later at Line 9.

Removing a U at v_4 in Fig. 8 can only introduce a new V or U at w_2, v_4, or one of their descendants, because all other nodes have unchanged source-sink path lengths and no fewer Steiner nodes on their source-sink paths. The subroutine **Clean_Up** checks for V's and U's at w_2 and v_4, and recursive calls to **Clean_Up** will eventually terminate because a new V or U can be introduced only by reducing the number of nodes on the n_0-v_4 path.

Fig. 12 shows how **Clean_Up** can require several recursive calls before terminating. However, for any node v', a call to **Remove_U** (v') will introduce a new V or U at v' or *parent* (v') only if it reduces the number of nodes on the n_0-v' path. Because any Steiner tree connecting $k+1$ points can contain at most $2k$ nodes in total, there are $O(k)$ nodes on the n_0-v' path. Hence, at most $O(k)$ calls can be made to **Clean_Up** for each node v' added to the queue in Line 6, and the total number of calls to **Clean_Up** is $O(k^2)$. □

Appendix B
Optimal Steiner ERT's

For minimum-cost Steiner trees, the classic result of [17] restricts the choice of Steiner nodes to at most $|N| \cdot |N-1|$ points (the "Hanan grid") and enables finite branch-and-bound methods to determine optimal solutions. Here, we prove an analogous result for trees minimizing any weighted average of sink Elmore delays. Like Hanan, we show that any tree containing a Steiner node which is not a vertex in the Hanan grid can have its edges and Steiner nodes shifted to lie on the Hanan grid. However, we do not shift edges in the same way as Hanan (the edge shifts he uses can be suboptimal in terms of Elmore delay). Indeed, the result of, e.g., Lemma B1 below is obvious when minimizing tree cost, but requires a fairly involved proof when minimizing Elmore delay. Our development of the Hanan grid result becomes complete with the proof of Lemma B4 below. In Lemma B5, we extend our result to show that the branch-and-bound SORT-C method

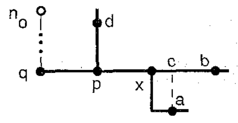


Fig. 13. Proof of Lemma B1: Node $a \in T^*$ is connected to edge $(p,b) \in T^*\backslash a$ at node x; either $x = p = n_0$ or $x = c$, where c is the closest connection between a and (p,b).

described in Section IV-D returns the optimal delay Steiner tree.[21]

A. Definitions

We assume that all delays are defined in terms of Elmore delay. We seek to characterize the *optimal* Steiner tree over N, denoted by T^*, which minimizes the weighted sum of sink delays $f = \sum_{i=1}^{k} \alpha_i \cdot t(n_i)$, with each $\alpha_i > 0$. (The case of some $\alpha_i = 0$ is effectively handled by setting these α_i to a small positive value.) We assume that T^* contains no Steiner nodes with degree < 3. For convenience, we normalize time and distance so that unit wire resistance and unit wire capacitance are both equal to one. We also consider a tree to be defined as a set of nodes and edges, so that the notations $v \in T$ for node v and $e \in T$ for edge e are well defined. An edge that is completely vertical or horizontal is called a *straight edge*; any other edge is called an *L-shaped edge*.

The *closest connection* between three nodes is the location of the single Steiner node in a minimum-cost Steiner tree over the three nodes. This location is unique and has coordinates given by the medians of the x- and the y-coordinates of the three nodes (if the minimum-cost Steiner tree is a chain, then the closest connection is the middle node). The *closest connection* between a node v and an edge e is the closest connection between v and the two endpoints of e. Assume that a Steiner tree T over N is rooted at n_0. We define $T\backslash v$ to be the tree induced by removing node v and all of its descendants from T, and then removing all degree-2 Steiner nodes from the resulting tree. We say that node $v \in T$ is *connected to* an edge $e \in T\backslash v$ if its parent node in T is located on edge e. If $parent(v)$ is located at the closest connection between v and an edge $e \in T\backslash v$ to which v is connected, then v is said to make a *closest connection* to e in T.

B. Proof of Closest Connections in T^*

Lemma B1: Let x be the parent of node $a \in T^*$, $a \neq n_0$. Then either $x = n_0$, or else x is located at the closest connection between a and each edge in $T^*\backslash a$ that is incident to a in T^*.

Proof: (See Fig. 13.) Let $e = (p,b) \in T^*\backslash a$ be an edge to which a is connected at node x in T^*. Let c be the location of the closest connection between a and e. Assume that node p has degree three (the proof is nearly identical if p has degree four), and let q be p's parent and d be the other child of p besides x. In our proof, we also assume that $p \neq n_0$ and that

[21] The following clarifications should be made about our results. First, we allow the source pin n_0 to have degree > 4, which is in general physically impossible, but can be approximated by merging wires close to the source. Second, the optimal delay Steiner trees will not always be planar, as this is not required by our definition of an optimal-delay tree.

p is the closest point to a on edge edge (q,p). The other cases are handled easily by analogous proofs.[22]

For convenience, we overload x, a, b, c, and p to also represent the respective path lengths from q to these nodes or locations. Even though c is not necessarily a node in T^*, we use T_c^* to represent the subtree of T^* below location c. Finally, we use C_a, C_b, and C_c to represent the tree capacitance in subtrees T_a^*, T_b^*, and T_c^*, respectively.

It is easy to see that $x \le c$, since otherwise moving x forward to c will reduce or leave unchanged all subtree costs (i.e., capacitance terms) and all path lengths (i.e., resistance terms). We will show that delay function f is concave in terms of x for $0 \le x \le c$. Our proof invokes several facts from elementary analysis: 1) any concave function defined over a real interval will be minimized at one of the two end points of the interval; 2) multiplying any concave function by a positive constant also gives a concave function; 3) the sum of two concave functions is also concave; and 4) any quadratic function of x with a negative coefficient for x^2 is concave in terms of x.

Consider the contribution made by the edge (x,a) to Elmore delay at various sinks $n_j \in T^*$. First, consider the case of $n_j \in T_a^*$. Delay $t(n_j)$ is the sum of four functions: f_1 = delay from n_0 to p; f_2 = delay from p to x due to capacitance in $T^*\backslash b$; f_3 = delay from p to x due to capacitance in edge (b,q) and T_b^*; and f_4 = delay from x to n_j. Simple application of the Elmore formula for these four functions gives

$$f_1 = K_0 + K_1(K_2 + a - x) \tag{2}$$

$$f_2 = x * \left(\frac{x}{2} + a - x + C_a + c - x + C_c\right) \tag{3}$$

$$f_3 = x * (b - q + C_b) \quad \text{if } x \le q \tag{4}$$

$$f_3 = q * (b - q + C_b) \quad \text{if } x \ge q \tag{5}$$

$$f_4 = (a - x) * \left(\frac{a - x}{2} + C_a\right) + K_3 \tag{6}$$

where K_0, K_1, K_2 and K_3 are constants. To be precise, K_0 is the sum of resistance/capacitance products along the n_0-p path; K_1 is the sum of resistances from n_0 to p; K_2 equals the total capacitance in T_p^* minus the edge (x,a); and K_3 is the delay from a to n_j. Function f_1 is linear in x, while f_2 and f_4 are quadratic in x. The equation for $f_2 + f_4$ has a negative coefficient for x^2, and so $f_2 + f_4$ is concave. Function f_3 is linear and increasing for $x \le q$ and remains constant for $x \ge q$; thus, f_3 is also concave in x. Consequently, $t(n_j) = f_1 + f_2 + f_4 + f_3$ is concave in x.

If $n_j \in T_c^*$ then f_1, f_2, and f_3 are identical to the case of $n_j \in T_a^*$. Function f_4 equals $(c-x)*(\frac{c-x}{2}+C_c)+K_2$, where K_2 is the delay from c to n_j. Again, f_1, f_3, and $f_2 + f_4$ are each concave in x, and so $t(n_j)$ is concave in x.

If $n_j \in T_b^*$ or $n_j = q$, we can express delay to n_j in terms of three functions f_1, f_2 and f_3. The definitions of f_1 and f_2 are the same as for $n_j \in T_a^*$, and f_3 gives the delay from p to n_j due to capacitance in $T^*\backslash a$. The equation for f_1 is identical to that for $n_j \in T_a^*$, while f_3 is a constant in terms of x. Hence, f_1 and f_3 are concave. For f_2, we have

$$\begin{aligned} f_2 &= x * (a - x + C_a) \quad \text{if } x \le q & (7)\\ &= q * (a - x + C_a) \quad \text{if } x \ge q. & (8)\end{aligned}$$

Any continuous, piece-wise differential function of a real variable is concave as long as its first derivative is monotone decreasing. It is clear that this property holds for f_2, except possibly at $x = q$. Let f_2' be the derivative of f_2. Then for $x < q$, $f_2'(x) = a - 2x + C_a$, and for $x > q$, $f_2'(x) = -q$. Substituting q for x in these equations, we see that f_2' is indeed decreasing at $x = q$ (because $a > x$). Consequently, f_2 is concave in x and so is $t(n_j)$.

Delay to any other sink in T^* is linear (and thus concave) in x. Therefore, because f is a nonnegative linear combination of concave functions over the interval $0 \le x \le c$, it is also concave over this interval and will be minimized at $x = 0$ or $x = c$. However, we assumed that a is connected to (p,b) and so $x \ne q = 0$. The only exception occurs if p has no parent, i.e., $p = n_0$. Since $e = (p,b)$ is an arbitrary edge in $T^*\backslash a$ to which a is connected, it must be that a makes a closest connection to *any* edge it is connected to (unless a's parent is n_0). □

[22] If $p = n_0$, then a similar argument shows that f is concave between $p = n_0$ and c, and will be minimized at one of these two points. If some point on (q,p) is closer to a than p, then a similar proof shows concavity for f over the interval between p and c. In this case, it is easy to show $x \ne p$ because connecting a to a closer point on (q,p) produces a lower value for f than connecting it to p, and so x must be located at c.

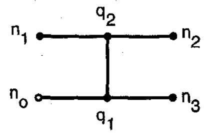


Fig. 14. Example of a routing tree T which cannot be constructed by algorithm BB-SORT-C, but which satisfies the condition that each nonsource node $v \in T$ makes a closest connection to each incident edge in $T\backslash v$.

Straightforward corollaries of Lemma B1 include: 1) any nonsource node in the optimal delay tree T^* must have degree $\le$ 4, and 2) the possible configurations of edges incident to a Steiner node $q \in T^*$ are restricted to the five configurations shown in Fig. 22. Note that Lemma B1 by itself is not sufficient to prove that BB-SORT-C will return the optimal delay tree. For example, if T^* connects a four-pin net into an "H" with two degree-3 Steiner nodes q_1 and q_2 (see Fig. 14), then the parent of each nonsource node v is connected by a closest connection to $T^*\backslash v$. However, T^* cannot be constructed by BB-SORT-C since the "H" cannot be formed by adding the three sinks sequentially by closest connections to the growing tree.

C. Hanan Grid Proof for Steiner Nodes in T^*

We root any routing tree T at the source n_0 and for any node $v \in T$ define T_v to be the subtree of T rooted at v. We define a *segment* to be a contiguous set of straight edges in T which are either all horizontal or all vertical; a *maximal segment* (MS) is a segment not properly contained in any other segment. Let M be an MS in T. The node in M closest to n_0 on a source-sink path containing M is called the *entry point* to M. A segment containing all points in M to one side of a node v located on M is a *half segment* with respect to v, and a half segment with

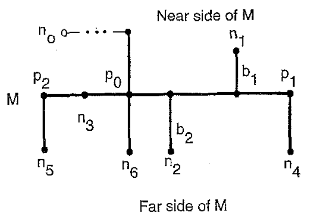


Fig. 15. Example of a maximal segment M with entry point p_0, one near branch b_1, and four far branches, including b_2. Note that by definition, n_3 forms a far branch with no edges. Also, edge (p_0, n_6) does *not* form a far branch off of M because p_0 is not an entry point to the MS containing (p_0, n_6).

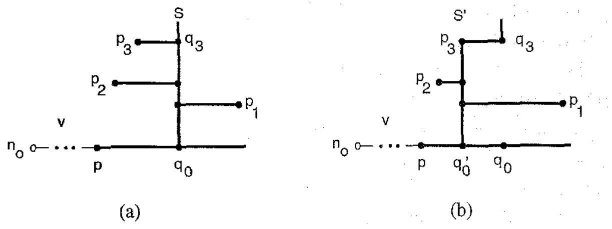


Fig. 16. Proof of Lemma B2: Example (a) with $|Near(S)| > |Far(S)|$ for a segment S between q_0 and q_3; (b) shows how S can be shifted to S' to reduce delay to all sinks in T^*_{q0} and leave delay unchanged at all other sinks.

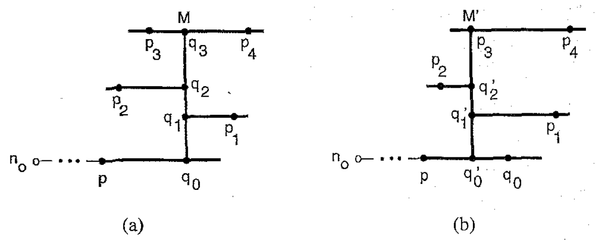


Fig. 17. Proof of Lemma B3: (a) Example where $|Near(M)| = |Far(M)|$for maximal segment M. M can be shifted to M', as shown in (b), to reduce delay at all sinks in the subtree T^*_{q0}.

respect to the entry point of M is called a *branch.* A branch b is called a *branch off of* MS M' if M' contains b's entry point and is perpendicular to b. Note that any given segment, M, will divide the plane into two half-planes. If M does not contain n_0, then the half-plane containing the edge between M's entry point and its parent is called the *near side* of M (because it is "nearer" to the source), and the other half-plane is called the *far side* of M. (If M contains the source, the near and far sides of M can be labeled arbitrarily.) Branches off of M that are located on its near (resp. far) side are called *near* (resp. *far*) *branches.* In addition, a *sink* located on M is defined to be a *far branch* off of M if none of its children are located on the far side of M (i.e., it is not the entry point to a larger far branch). For any segment S, we use $Near(S)$ (resp. $Far(S)$) to denote the set of near (resp. far) branches off of the maximal segment containing S. Fig. 15 gives an example of an MS M with endpoints p_1 and p_2, entry point p_0, and four branches, including near branch b_1, far branch b_2, and a far branch consisting only of sink n_3.

Lemmas B2 and B3 establish some properties that must hold for any maximal segment in T^*. Lemma B4 then uses these properties to show that each maximal segment in T^* will have a sink located on it. An immediate corollary of Lemma B4 is a generalization of the classic result of Hanan [17] to the Elmore delay objective. (Hanan's original theorem may be viewed as a special case of this Corollary with the driver on-resistance $r_d \to \infty$.)

Lemma B2: In the optimal tree T^*, let q_0 be the entry point to a maximal segment M not containing n_0. Let S be any segment contained in M and having q_0 as an endpoint. Then $|Far(S)| \geq |Near(S)|$.

Proof: By contradiction. Let S be the smallest segment in M with q_0 as an endpoint so that $Near(S) > Far(S)$. Then a portion of T^* between n_0 and q' looks like Fig. 16(a). Label the branches $b_1, \ldots, b_j$ in order from entry point q_0. Fig. 16(b) shows how we can shift segment S topologically toward the source; this effectively shifts wire from each near branch to a far branch with is topologically closer to the source (i.e., with a smaller label). Shifting S does not affect tree cost,[23] and source-sink path lengths will be unchanged to all sinks except those connected to the tree through branches in $Near(S)$, which will have reduced source-sink path lengths. Consequently, the shift will decrease delay to all sinks below q_0 in T^* and leave delay to all other sinks unchanged, contradicting the optimality of T^*. $\square$

[23] Unless q_0 is an endpoint for the MS containing edge (p, q_0), in which case tree cost will decrease.

We can now use Lemma B2 to show that if an MS M has as many near branches as far branches, then it can be shifted so as to reduce delay to some sinks and leave delay to others unchanged. This is because shifting M toward the source will not increase total wirelength and will decrease some source-sink pathlengths and shift capacitance along some source-sink paths closer to the source. Intuitively, this is the proof behind our next lemma.

Lemma B3: In the optimal tree T^*, let M be an MS not containing n_0. Then $|Far(M)| > |Near(M)|$.

Proof: By Lemma B2, $|Far(M)| \geq |Near(M)|$. Suppose that the exact equality $|Far(M)| = |Near(M)|$ holds. Lemma B2 then implies that each endpoint of M has a near branch incident to it as in Fig. 17(a) (otherwise, M would contain a subsegment S with all but one endpoint of M and having $|Near(S)| > |Far(S)|$.) In Fig. 17(b), we show how M can be shifted toward the source without increasing total wirelength, while reducing source-sink pathlengths to nodes on the near branches of M and shifting capacitance toward the source for nodes on the far branches of M. Consequently, moving M will reduce delay to all sinks in T^*_{q0} and leave delay to all other sinks unchanged (or reduced if the shift reduces total wirelength), thereby contradicting the optimality of T^*. $\square$

Lemma B4: In the optimal tree T^*, any maximal segment must contain either the source or a sink.

Proof: (See Fig. 18.) Let M be a lowest maximal segment in T^* which does not contain either the source or a sink, i.e., every MS that is topologically below M contains a sink. Let q_0 be the entry point to M and let p_0 be the parent node of q_0 in T^*. Consider the possibility of shifting M, either toward

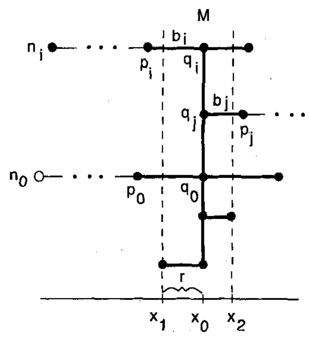


Fig. 18. Proof of Lemma B4: Because the objective function f is concave in r over the interval $0 \leq r \leq x_2 - x_1$, f will be minimized when the maximal segment M passes through either the gridline at x_1 or the gridline at x_2.

the source or away from the source, without passing over any node in T^*_{q0} which is not in M. Without loss of generality, assume that M is a vertical segment with x-coordinate x_0, and with the near side of M having $x < x_0$. Let $x_1 < x_0$ be the closest value to x_0 on the near side of M such that shifting M to $x = x_1$ would cause M to intersect a node that is in T^*_{q0} but not in M. Similarly, let $x_2 > x_0$ be the closest value to x_0 on the far side of M such that shifting M to $x = x_2$ would cause M to intersect some node that is in T^*_{q0} but not in M. Let the variable r, $0 \leq r \leq x_2 - x_1$, denote the position of M between the x-coordinates x_1 and x_2. We will show that minimizing the delay function f implies that either $r = 0$ or $r = x_2 - x_1$.

Let $d = Far(M) - Near(M)$. Consider the delay to some sink n_i located along a near branch b_i off of M which has entry point q_i. (In general, we let q_j denote the entry point to branch b_j.) Delay $t(n_i)$ is quadratic in r only along the edge (p_0, q_0) and along the edge (q_i, p_i), where p_i is the child of q_i on b_i. To be precise, the delay due to (p_0, q_0) is equal to $r * (r/2 - d * r + K)$, where K is some constant; the delay due to (q_i, p_i) is equal to $r * (r/2 + K') + K''$, where K' and K'' are again constants. Therefore, the equation for $t(n_i)$ is

$$t(n_i) = (1 - d) * r^2 + K_1 * r + K_0 \tag{9}$$

where K_1 and K_0 are constants. From Lemma B3, we know that $d \geq 1$, implying that $t(n_i)$ is a concave function of r. Similarly, delay to a sink n_j along a far branch b_j off of M will be equal to

$$-d * r^2 + K_1' r + K_0' \tag{10}$$

where again K_1' and K_0' are constants; this too is a concave function of r. Finally, delay to any sink whose source-sink path does not contain an edge in M will be linear in r, and thus also a concave function. Since any linear combination of functions that are each concave on a given interval will also be concave on that interval, f is concave in r and is minimized at one of its extreme values, i.e., at $r = 0$ or $r = x_2 - x_1$.

Thus, M may be moved so that it contains a new node, say p_i. If p_i is a sink, the lemma is proved. If p_i is a Steiner node, then because it has degree > 2, there must be a vertical MS incident to p_i, and this vertical MS must contain a sink since M is the lowest maximal segment not containing a sink. Hence, if p_i is a Steiner node, the shifted M will also contain a sink. □

A direct corollary of Lemma B4 is that all Steiner nodes in the Elmore-optimal Steiner tree are contained in the Hanan grid.

Corollary: Let X be the set of x-coordinates for all pins in N, and let Y be the set of y-coordinates in N. Then if (x, y) is the location of a Steiner node in T^*, $x \in X$ and $y \in Y$.

Thus, only a finite number of possible Steiner point locations need to be considered. Hanan's original theorem may be viewed as a special case of this Corollary with the driver on-resistance $r_d \rightarrow \infty$.

D. Decomposition Theorem for T^*

To prove that BB-SORT-C will return the optimal-delay tree T^*, we show that T^* can be constructed by starting with a tree T_0 containing only n_0, then adding a sequence of sinks n_i, $1 \leq i \leq k$, each of which yields a tree T_i by making a closest connection to some edge in the current tree T_{i-1}. We show that such a sequence of trees exists by starting with $T^* = T_k$ and $i = k$, then "peeling off" an n_i at each iteration such that n_i is joined by a closest connection in T_i to some edge in $T_{i-1} = T_i \backslash n_i$.

At each step, we find an interior node $q \in T_i$ whose children are all leaves. Each of these leaves must be a sink, because all low-degree Steiner nodes (i.e., with degree < 3) are removed from $T_{i+1} \backslash n_{i+1}$. We choose one of q's leaves to be the n_i that is peeled off, and set $T_{i-1} = T_i \backslash n_i$. The choice of which leaf should be peeled is guided by the function $Pin(q)$, which specifies one of q's children that should *not* be peeled off from q. Thus, when q is removed as a low-degree Steiner node, the edge between q and its parent is replaced with an edge between *Pin* (q) and q's parent. More formally, we define *Pin* (q) for each node $q \in T^*$ as follows: 1) if q is the source or a sink, then *Pin*$(q) = q$; and 2) if q is a Steiner node, then *Pin*(q) is chosen according to the template given in Fig. 19.

After *Pin*(q) has been assigned, we can apply the rules described in Fig. 20 to peel off sinks, thus determining the sequence in which sinks can be added to construct T^*. Note that node p in Line 3 of Fig. 20 must exist since T_i is finite and has no cycles. Fig. 21 gives an example of the decomposition procedure applied to an eleven-pin net. Sinks in the figure are labeled in reverse order of how the decomposition procedure might peel them off from the tree. (Other orders are possible because the decomposition procedure is not completely deterministic.) Table XII shows how *Pin*(q) was assigned for each Steiner node in Fig. 21.

We now show that the procedure of Fig. 20 gives a sequential decomposition of the optimal-delay tree T^*, such that each T_i is constructed by connecting sink n_i to tree T_{i-1} by a closest connection to some edge in T_{i-1}.

Lemma B5: There exists a sequence of subtrees $T_0 = \{n_0\}, T_1, T_2, \ldots, T_k = T^*$ such that for each i, $1 \leq i \leq k$, 1) there is a sink $n_i \in T_i$ such that $T_{i-1} = T_i \backslash n_i$, and 2) either n_i is connected to n_0, or n_i makes a closest connection in T_i to some edge in T_{i-1}.

```
Pin(q) Assignment Procedure
Input: Optimal delay tree $T^*$
Steiner noder $q \in T^*$ such that Pin(q) has been assigned
for each $w \in T_q^*, w \neq q$
Output: Pin(q)
1.  $p$ = parent(q) in $T^*$
2.  If edge $(p, q)$ is a straight edge
3.      Set $M$ to be the MS containing $(p, q)$
4.  If edge $(p, q)$ is L-shaped
5.      Set $c$ arbitrarily to be one of $q$'s two children
            /* ($q$ has exactly two children, by Lemma B1) */
6.  Else if $T_q$ contains a sink on $M$
7.      Set $c$ to be a child of $q$ on $M$
8.  Else if $p$ is the entry point to $M$
9.      Set $B$ to be the far branch of $M$ at $q$
            /* (such a $B$ exists by Lemma B2) */
10.     Set $c$ to be a child of $q$ on $B$
11. Else if $p$ has degree 4
12.     Set $c$ arbitrarily to be one of $q$'s children
13. Else if there is a near (far) branch of $M$ at p
14.     If there is a far (near) branch $B$ of $M$ at $q$
15.         Set $c$ to be the child of $q$ on B
16.     Else Set $c$ to be the child of $q$ on M
17. Pin(q) = Pin(c)
```

Fig. 19. Criteria used to associate a sink $Pin(q)$ with each Steiner node q in the optimal-delay tree T^*. The assignment determines which sink in T_q will remain in the tree when q is removed from the current tree while "peeling off" sinks from T^*.

```
$T^*$ Decomposition Procedure
Input: Optimal delay tree $T^*$
Output: Sequence of sinks $n_1, \ldots n_k$ used to construct $T^*$
        using only closest connections of each $n_i$ to $T_{i-1}$
1.  $i = k$
2.  Repeat until $i == 0$
3.      Find a node $q \in T_i$ whose children are all leaves
4.      If $q == n_0$ set $c$ to be any child of $n_0$ in $T^*$
5.      Else if $q$ has degree 4
6.          Set $c$ to be the child of $q$ in $T^*$ on the MS
            containing edge $(parent(q), q)$
7.      Else Set $c$ to be a child of $q$ such that $Pin(c) \neq Pin(q)$
8.      $n_i = Pin(c)$
9.      $T_{i-1} = T_i \backslash n_i$
10.     $i = i - 1$
```

Fig. 20. Procedure to determine a sequence of sinks $n_1, \ldots, n_k$ which can be used to construct T^* by a sequence of closest connections from n_i to tree T_{i-1}.

Proof: Part 1) of the Lemma is true since the construction of Fig. 20 removes exactly one sink during each pass through Lines 3 to 9.

To show 2), let p be the parent of the node q at Line 3 in Fig. 20. The first case is when q is a sink or a degree-4 Steiner node in T_i [as in Fig. 22(e)]. In this case, edge (p, q) will remain in tree T_{i-1}. If (p, q) is L-shaped, we must have a connection as in Fig. 22(a), where the two children of q are eventually replaced by sinks on the maximal segments with entry point q (i.e., n_1 and n_2 in the Fig. 22). Both of these sinks have closest connections to (p, q) at q. If (p, q) is a straight edge, let M be the MS containing (p, q), and let a be a child of q in T^*. The sink $Pin(a)$ is assigned in the Fig. 19 template such that the q–$Pin(a)$ path in T^* will contain only edges in M, edges in branches off of M, or edges in a sequence of far branches off of branches of M. (For example, consider the paths from q to sinks n_1 and n_2 in Fig. 22(c)–(e).) Thus, $Pin(a)$ and p cannot be on the same side of a line that passes

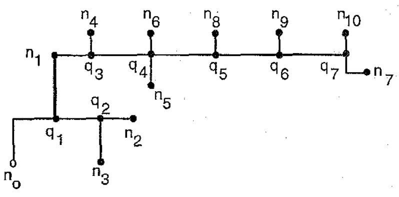


Fig. 21. Example of the order in which pins are "peeled" from an optimal-delay Steiner tree T^*. (Sinks n_i are peeled from T^* in reverse order of their subscripts.)

TABLE XII
PIN ASSIGNMENT TO STEINER NODES IN THE EXAMPLE OF FIG. 21. LINE NUMBERS REFER TO THE CORRESPONDING LINE IN THE PIN ASSIGNMENT PROCEDURE OF FIG. 19

Pin Assignments for Example Tree		
Steiner node q	$Pin(q)$	Reason for Assignment
q_1	n_1	(n_0, q_1) is L-shaped (Line 4)
q_2	n_2	n_2 is on MS containing (q_1, q_2) (Line 6)
q_3	n_4	n_1 is the entry point to MS containing (n_1, q_3) (Line 8)
q_4	n_5	far branch at q_3 and near branch at q_4 (Line 15)
q_5	$Pin(q_6)$	q_4 has degree 4 (Line 11)
q_6	$Pin(q_7)$	far branch at q_5 and no near branch at q_6 (Line 16)
q_7	n_7	far branch at q_6 and near branch at q_7 (Line 15)

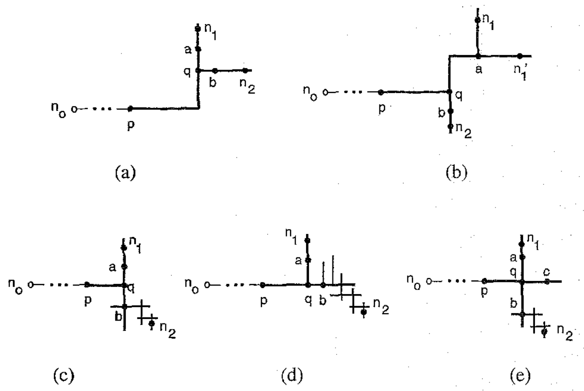


Fig. 22. Five possible topologies at any Steiner node q in T^*. Each diagram shows two sinks n_1 and n_2 below node q in the tree, such that q is the closest connection between n_1, n_2 and q's parent p.

through q and is perpendicular to M. Consequently, q will be the closest connection between edge (p, q) and $Pin(a)$.

The second case is when q is a degree-3 Steiner node in T_i. Let a and b be the children of q in T^* such that $Pin(a)$ and $Pin(b)$ are q's children in T_i. Without loss of generality, we assume that $Pin(q) = Pin(a)$ and $n_i = Pin(b)$. We must show that q is located at the closest connection between nodes p, $Pin(a)$, and $Pin(b)$. There are four possible configurations for connections at q, as shown in parts Fig. 22 (a)–(d).

- In Fig. 22(a), edge (p, q) is L-shaped and both $Pin(a)$ and $Pin(b)$ (denoted by n_1 and n_2 in the figure) must be on maximal segments with entry point q; it is easy to see that q is the closest connection between p, $Pin(a)$, and $Pin(b)$.

In Fig. 22(b)–(d), edge (p, q) is a straight edge. Let M be the MS containing (p, q), and let M' be the MS perpendicular to M with entry point q.

- In Fig. 22(b), edge (q, a) is L-shaped and edge (q, b) is on the MS M'. By Lemma B4, M' must contain a sink, which will be contained in subtree T_b. Thus, $Pin(b)$ (n_2 in the figure) is located on M'. Node a is the entry point for two perpendicular branches each containing sinks (by Lemma B4); *Pin*(a) is chosen arbitrarily from one of these branches (Line 3 in Fig. 19). In Fig. 22, either *Pin*$(a) = n_1$ or *Pin*$(a) = n'_1$; thus, it can be seen from the figure that q is the closest connection between p, *Pin*(a), and *Pin*(b).

- In Fig. 22(c), M' is the union of two branches. One of these branches contains a sink (by Lemma B4); without loss of generality, let this be the branch containing edge (q, a), with *Pin*$(a) = n_1$ in M'. Let B be the branch containing edge (q, b). If *Pin*(b) is on B, then q will be the closest connection between p, *Pin*(a) and *Pin*(b). Otherwise, according to Lemma B2 we must have that b is the entry point to a far branch off of M'. Hence, if *Pin*(b) is not on B, the b—*Pin*(b) path in T^* contains only edges on far branches [by the criteria in Lines 8-10 in Fig. 19; see $n_2 =$ *Pin*(b) in Fig. 22(c)]. Thus, *Pin*(b) is contained in the upper-right quadrant relative to q in the figure, and q is the closest connection between p, *Pin*(a), and *Pin*(b).
- Finally, consider the configuration in Fig. 22(d). Here, MS M' is a branch of M containing node a and sink *Pin*(a). Suppose that M' is a far branch; if *Pin*(b) is not on MS M, then there must be a near branch off of M somewhere below q in T^* (otherwise, we could reduce all delays by shifting the entire half segment of M below q toward a). Let B_j be the near branch below q closest to q. Either sink *Pin*(b) is on B_j, or the q_j–*Pin*(b) path in T^* consists only of edges in B_j or far branches. In either case, *Pin*(b) ($= n_2$ in the figure) is contained in the lower-right quadrant relative to q. If M' is a near branch, an analogous argument again shows that *Pin*(b) is in q's lower-right quadrant. Thus, q is the closest connection between p, *Pin*(a) and *Pin*(b). □

Except for redundancies and pruning of suboptimal trees, BB-SORT-C searches over all possible ways to construct a Steiner tree sequentially, such that each sink is added by a closest connection to some edge in the current tree. Thus, we have

Theorem B1: For any positive linear combination of sink delays, $f = \sum_{i=1}^{k} \alpha_i \cdot t(n_i)$, $\alpha_i > 0\ \forall i$, algorithm BB-SORT-C returns a Steiner tree T^* which minimizes f.

Acknowledgment

The authors are grateful to the authors of [38] for use of their Two-Pole simulator code. E. S. Kuh and K. Chaudhary provided many helpful comments regarding modeling and experimental methodology. A. Vittal of the University of California, Santa Barbara gave many helpful comments and criticisms on the proofs in an earlier draft, and we also thank the anonymous referees for many detailed comments. J. Aylor and M. Shur, and S. Muddu, provided invaluable assistance with SPICE simulation methodology. The hospitality of E. S. Kuh and his research group is gratefully acknowledged. Additional related papers may be found at `http://www.cs.virginia.edu/~robins/` and `http://ballade.cs.ucla.edu:8080/~abk/`.

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," University of California, Los Angeles, Dept. Computer Science, Tech. Rep. CSD-920051, 1992.

[2] T. G. Andrews, Ed., *Methods of Psychology.* New York: Wiley, 1948.

[3] B. Awerbuch, A. Baratz, and D. Peleg, "Cost-sensitive analysis of communication protocols," in *Proc. ACM Symp. Principles Distributed Computing*, 1990, pp. 177–187.

[4] T. Barrera, J. Griffith, G. Robins, and T. Zhang, "Narrowing the gap: Near-optimal Steiner trees in polynomial time," in *Proc. IEEE Int. ASIC Conf.*, Rochester, NY, Sept. 1993, pp. 87–90.

[5] K. D. Boese, J. Cong, A. B. Kahng, K. S. Leung, and D. Zhou, "On high-speed VLSI interconnects: Analysis and design," in *Proc. Asia-Pacific Conf. Circuits Syst.*, Sept. 1992, pp. 35–40.

[6] K. D. Boese, A. B. Kahng, and G. Robins, "High-performance routing trees with identified critical sinks," in *Proc. ACM/IEEE Design Automat. Conf.*, June 1993, pp. 182–187.

[7] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Fidelity and near-optimality of Elmore-based routing constructions," in *Proc. IEEE Int. Conf. Computer Design*, Oct. 1993, pp. 81–84.

[8] D.-S. Chen and M. Sarrafzadeh, "A wire-length minimization algorithm for single-layer layouts," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1992, pp. 390–393.

[9] J. P. Cohoon and L. J. Randall, "Critical net routing," *Proc. IEEE Int. Conf. Computer Design*, 1991, pp. 174–177.

[10] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 739–752 June 1992.

[11] J. Cong, K.-S. Leung and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," in *Proc. ACM/IEEE Design Automat. Conf.*, 1993, pp. 606–611.

[12] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[13] W. E. Donath, *et al.*, "Timing driven placement using complete path delays," in *Proc. ACM/IEEE Design Automat. Conf.*, 1990, pp. 84–89.

[14] A. E. Dunlop, V. D. Agrawal, D. N. Deutsh, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip layout optimization using critical path weighting," in *Proc. ACM/IEEE Design Automat. Conf.*, 1984, pp. 133–136.

[15] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55–63, 1948.

[16] S. Even, *Graph Algorithms.* Potomac, MD: Computer Science, 1979.

[17] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl. Math.*, vol. 14, pp. 255–265, 1966.

[18] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit placement for predictable performance," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1987, pp. 88–91.

[19] J.-M. Ho, G. Vijayan, and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 2, pp. 185–193, 1990.

[20] M. A. B. Jackson and E. S. Kuh, "Estimating and optimizing RC interconnect delay during physical design," in *Proc. IEEE Int. Conf. Circuits Syst.*, 1990, pp. 869–871.

[21] M. A. B. Jackson, E. S. Kuh, and M. Marek-Sadowska, "Timing-driven routing for building block layout," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 518-519, 1987.

[22] A. B. Kahng and G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 893–902, July 1992.

[23] S. Khuller, B. Raghavachari, and N. Young, "Balancing minimum spanning and shortest path trees," *Proc. ACM/SIAM Symp. Discrete Algorithms*, to appear.

[24] S. Kim, R. M. Owens, and M. J. Irwin, "Experiments with a performance driven module generator," in *Proc. ACM/IEEE Design Automat, Conf.*, 1992, pp. 687–690.

[25] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* Berlin: Wiley-Teubner, 1990.

[26] I. Lin and D. H. C. Du, "Performance-driven constructive placement," in *Proc. ACM/IEEE Design Automat. Conf.*, 1990, pp. 103–106.

[27] M. Marek-Sadowska and S. Lin, "Timing driven placement," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1989, pp. 94–97.

[28] S. Prasitjutrakul and W. J. Kubitz, "A timing-driven global router for custom chip design," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1990, pp. 48–51.

[29] A. Prim, "Shortest connecting networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, pp. 1389–1401, 1957.

[30] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, "The rectilinear Steiner arborescence problem," *Algorithmica*, vol. 7, pp. 277–288, 1992.
[31] G. Robins, "On optimal interconnections," Ph.D. dissertation, Dept. Computer Science,, Univ. California, Los Angeles, June 1992.
[32] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD–2, no. 3, pp. 202–211, 1983.
[33] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "RITUAL: A performance driven placement algorithm for small-cell IC's," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 48–51.
[34] S. Sutanthavibul and E. Shragowitz, "Adaptive timing-driven layout for high speed VLSI," in *Proc. ACM/IEEE Design Automat. Conf.*, 1990, pp. 90–95.
[35] S. Teig, R. L. Smith, and J. Seaton, "Timing driven layout of cell-based IC's," *VLSI Syst. Design*, May 1986, pp. 63–73.
[36] R. S. Tsay, "Exact zero skew," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 336–339.
[37] D. Zhou, F. P. Preparata, and S. M. Kang, "Interconnection delay in very high-speed VLSI," *IEEE Trans. Circuits Syst.*, vol. 38, July 1991.
[38] D. Zhou, S. Su, F. Tsui, D. S. Gao, and J. Cong, "Analysis of trees of transmission lines," Univ. California, Los Angeles, Tech. Rep. CSD-920010, 1992.

Kenneth D. Boese (M'95) received the A.B. degree in mathematics from Princeton University, Princeton, NJ and received the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 1993 and 1995, respectively.

He is currently working for Cadence Design Systems in San Jose, CA. His research interests include VLSI global routing, and clock routing, cost surfaces in global optimization, multiagent iterative optimization, and high-level physical design for VLSI.

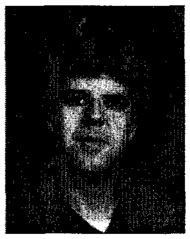
Dr. Boese is a member of the ACM.

Andrew B. Kahng (A'89), for a photograph and biography, see p. 1358 of the November issue of this TRANSACTIONS.

Bernard A. McCoy (M'95) received the B.A. degree in mathematics from Hiram College, Hiram, OH, and the M.S. degree in computer science from the University of Virginia, Charlottesville, in 1992 and 1994, respectively.

He is presently a Software Design Engineer at Nu-Mega Technologies, Nashua, NH. His research interests include VLSI CAD and automatic error-detection software, with recent work focusing on performance-driven routing algorithms.

Mr. McCoy authored the winning paper in the 1991 COMAP contest in mathematics, which was also a distinguished paper at the 1991 ORSA/SIAM Conference. He is a member of the ACM and MAA.



Gabriel Robins (S'91–M'91), received the Ph.D. degree in computer science from the University of California, Los Angeles in 1992.

Currently, he is an Assistant Professor of Computer Science in the University of Virginia, Charlottesville. He is a member of the Defense Science Study Group, an advisory board to the U.S. Department of Defense. His research interests are in VLSI CAD and geometric algorithms, with recent work focusing on performance-driven routing, pattern recognition, and computational biology. He is the author of a book on high-performance routing for Kluwer.

Dr. Robins is the author of a Distinguished Paper at the 1990 IEEE International Conference on Computer-Aided Design. He is General Chair of the 1996 Physical Design Workshop, and he also serves on the technical program committees of several other leading conferences. He has won a Distinguished Teaching Award at the University of California, Los Angeles. He has won an NSF Young Investigator Award, a Packard Fellowship, a Lilly Foundation Teaching Fellowship, and an All-University Outstanding Teacher Award at the University of Virginia. He is a member of ACM, MAA, and SIAM.