# Non-Tree Routing*

Bernard A. McCoy and Gabriel Robins

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

## Abstract

*An implicit premise of existing routing methods is that the routing topology must correspond to a tree (i.e., it does not contain cycles). In this paper we investigate the consequences of abandoning this basic axiom, and instead allow routing topologies that correspond to arbitrary graphs (i.e., where cycles are admissible). We show that adding extra wires to an existing routing tree can often significantly improve signal propagation delay by exploiting a tradeoff between wire capacitance and resistance, and we propose a new routing algorithm based on this phenomenon. Using SPICE to determine the efficacy of our methods, we obtain dramatic results: for example, the judicious addition of a few extra wires to an existing Steiner routing reduces the signal propagation delay by an average of up to 62%, with relatively modest total wirelength increase, depending on net size and the technology parameters. Finally, we observe that non-tree routing also significantly reduces signal skew.*

## 1 Introduction

Recent advances in VLSI technology have steadily improved chip packing densities. As feature sizes decrease, device switching speeds tend to increase; however, thinner wires have higher resistance, causing signal propagation delay through the interconnect to increase [20]. Thus, interconnection delay has had a greater impact on circuit speed, being responsible for up to 70% of the clock cycle in the design of dense, high-performance circuits [22]. In light of this trend, performance-driven physical layout has become central to the design of leading-edge digital systems. Early work focused on performance-driven placement, with the usual objective being the close placement of cells in timing-critical paths [11] [16] [17].

Although timing-driven placement has a substantial effect on layout performance, the lack of optimal-delay interconnection algorithms impedes designers in fully exploiting a high-quality placement. Once a module placement has been fixed, good timing-driven interconnection algorithms are key to enhancing the performance of the layout solution. For a given signal net, the typical objective has been to minimize the maximum source-sink signal delay. Several approaches have appeared in the literature, e.g., Dunlop et al. [12] determine net priorities based on static timing analysis, and process higher priority nets earlier, using fewer feedthroughs; Jackson et al. [14] outline a hierarchical approach to timing-driven routing; and Prasitjutrakul and Kubitz [19] use an A* heuristic search and the Elmore delay formula [13] in their tree optimization; Cohoon and Randall [9] developed a critical net routing algorithm in order to reduce interconnect delay.

Cong et al. have proposed finding minimum spanning trees with bounded source-sink pathlength [10] by simultaneously minimizing both tree cost and the tree radius; another cost-radius tradeoff was achieved by Alpert et al. [1]. Boese et al. [6] have developed a "critical sink" routing approach which significantly reduces delay to specified sinks, thereby exploiting the critical-path information that is implicitly available during iterative timing-driven layout. Recently, Boese et al. [4] [5] have identified and exploited a high-quality, algorithmically tractable model of interconnect delay, based on an upper bound [21] for Elmore delay.

An implicit premise of previous methods is that a routing topology must correspond to a tree (i.e., an acyclic topology). In retrospect, this assumption seemed natural, since a tree topology spans a net and achieves electrical connectivity using a minimum number of edges. In this paper, we question this seemingly basic axiom, and investigate the consequences of allowing cycles in the routing. Thus, we formulate a routing problem where the interconnection topology may correspond to an arbitrary graph.

At this point, the reader may ask: how can adding extra wires to an existing routing tree possibly improve signal propagation delay? The answer lies in the tradeoff between the capacitance and resistance in a circuit. Clearly, adding extra wires to a routing tree increases the overall capacitance; however, the extra wires may significantly lower certain source-sink resis-

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

tance values. It is possible for this decrease in resistance to more than compensate for the associated increase in capacitance, especially for leading-edge technologies (a simple example of this phenomenon can be seen in Figure 1).

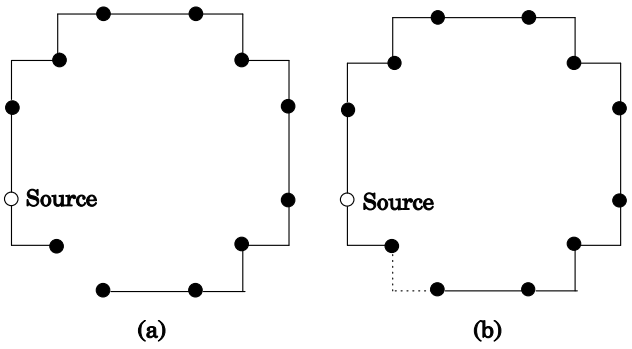


Figure 1: An example of how adding an extra edge to the minimum spanning tree on the left (a) can yield the routing topology with reduced interconnection delay on the right (b); in this example, routing topology (a) has maximum source-sink SPICE delay of 1.3ns, while the topology on the right has a SPICE delay of 1.0ns, a 23% improvement (at a total wirelength penalty of 9%). The interconnect parameters used are representative of a MOSIS 0.8$\mu$ CMOS process.

Since we are highly concerned with obtaining realistic results, we use the SPICE3e2 circuit simulator [18] to determine the efficacy of our methods. Our results are dramatic: adding wires/edges to an existing minimum Steiner tree routing reduces the signal propagation delay by an average of up to 62% over Steiner routing, while incurring only moderate wirelength penalties over Steiner routing, depending on net size and technology parameters.

Our algorithm is efficient, and our basic approach is amenable to numerous extensions of the routing design problem, such as critical sinks and wiresizing. In addition, non-tree routings offer increased *reliability* since, e.g., a single open fault along a wire loop will not cause a net to become disconnected. Thus, our methodology can tolerate certain types of faults (such as opens due to electro-migration in submicron technologies) with a graceful speed degradation, leaving circuit functionality intact. Non-tree routings can also improve signal skew characteristics by an average of up to 63% over Steiner routing, as well as reduce signal reflection [7].

## 2 Problem Formulation

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

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

The ORG problem generalizes the Optimal Routing Tree (ORT) problem of [4] [5], which corresponds to the special case where $G$ is a tree. As in [6], the ORG formulation easily extends to address critical sinks, by associating a *criticality* $\alpha_i \geq 0$ with each sink $n_i$, reflecting timing information obtained during the performance-driven placement phase. The goal would then be to construct a routing which minimizes the weighted sum of the sink delays $\sum_{i=1}^{k} \alpha_i \cdot t(n_i)$.

The specific routing graph $G$ that solves the ORG problem will depend on the model used to estimate the delay $t(G)$, as well as on the particular technology parameters. Ideally, we would like to compute and optimize delay according to the complete physical attributes of the circuit. To this end, we could use the circuit simulator SPICE [18], which is generally regarded as the best available tool for obtaining a precise, complete measure of interconnect delay.

Unfortunately, SPICE delay is too computationally prohibitive to evaluate during the routing phase of layout, and we are thus forced to seek other alternatives. Another delay model is the Elmore delay formula [13], which was shown in [4] [5] to have both high accuracy and fidelity in comparison with SPICE. The Elmore delay is defined as follows. Given routing tree $T$ rooted at $n_0$, let $e_i$ denote the edge from pin $n_i$ to its parent. The resistance and capacitance of edge $e_i$ are denoted by $r_{e_i}$ and $c_{e_i}$, respectively. Let $T_i$ denote the subtree of $T$ rooted at $n_i$, and let $c_i$ denote the sink capacitance of $n_i$. We use $C_i$ to denote the *tree capacitance* of $T_i$, namely the sum of sink and edge capacitances in $T_i$. Using this notation, the Elmore delay along edge $e_i$ is equal to $r_{e_i} \cdot (c_{e_i}/2 + C_i)$. Let $r_d$ denote the output driver resistance at the net's source. Then the Elmore delay $t_{ED}(n_i)$ from source $n_0$ to sink $n_i$ is given by:

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

We can extend the $t_{ED}$ function to entire trees by defining $t_{ED}(T) = \max_{i=1}^{k} t_{ED}(n_i)$. Because of its relatively simple form, Elmore delay can be calculated in $O(k)$ time [21]. However, while the basic Elmore delay model outlined above applies only to tree topologies, Chan and Karplus have extended it to RC meshes [8]. Their method partitions the graph into a spanning tree and a set of $m$ additional edges, then adds the extra edges back, updating the Elmore delay at each step. This increases the time complexity of the Elmore delay calculation to $O(k \cdot m)$. We use this method of delay calculation for general RC meshes in our approximation heuristic for the ORG problem.

## 3 Low Delay Routing Graph Heuristic

The ORG problem may be solved heuristically by starting with some reasonable tree topology such as a Steiner tree or a minimum spanning tree, and searching for some new edge to add, so that the delay in the resulting routing graph will be minimized. This edge is then added to the routing graph, and the process is iterated (i.e., we look for yet another good edge to add). We terminate when no further delay improvement is possible; thus, the maximum source-sink delay of the routing produced by our algorithm is guaranteed to be no worse than that of the initial routing (and typically considerably better). Steiner points are allowed as junctures in the routing, in order to afford further opportunity for both delay and wirelength optimization. An execution example of this method, called the Low Delay Routing Graph (LDRG) algorithm, is shown in Figure 3, while the formal statement of this algorithm is given in Figure 2.

| **Low Delay Routing Graph (LDRG) Algorithm** |
|---|
| **Input:** signal net $N$ with source $n_0 \in N$<br>**Output:** low-delay routing graph $G = (\hat{N}, E)$ |
| **Compute** a Steiner routing $G = (\hat{N}, E)$ over $\hat{N} = N \cup S$,<br>where $S$ are the possible Steiner points,<br>and $E \subseteq \hat{N} \times \hat{N}$ is the set of Steiner tree edges<br>**While** there is an edge $e_{ij} \in \hat{N} \times \hat{N}$<br>such that $t((\hat{N}, E \cup \{e_{ij}\})) < t(G)$ is minimized<br>**Do** $G = (\hat{N}, E \cup \{e_{ij}\})$<br>**Output** resulting routing topology $G$ |

Figure 2: The Low Delay Routing Graph algorithm.

## 4 Experimental Results

We implemented the LDRG algorithm using C in the UNIX/Sun environment; code is available from the authors. We ran trials on sets of 100 random nets for each of several net sizes; pin locations were chosen using a uniform distribution in a square layout region. Although the LDRG method uses the extension of the Elmore delay formula to graphs [8], for greater accuracy and realism, we used SPICE3e2 to determine the performance of routings produced by LDRG.

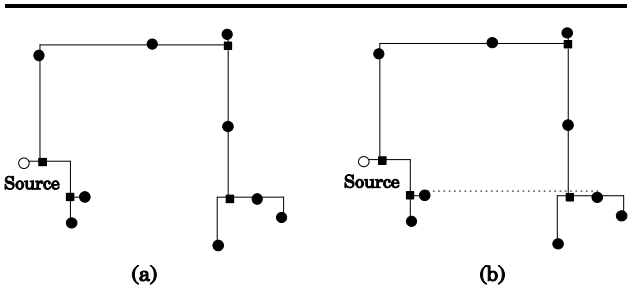


Figure 3: An execution of LDRG algorithm on a random 10-pin net. The Steiner tree shown on the left (a) has SPICE delay of 2.8ns (Steiner points are square), while the LDRG routing on the right (b) has SPICE delay of 1.9ns, a 32% improvement (the wirelength increase is 25%).

| Technology | IC1 $2.0\mu$ | IC2 $1.2\mu$ | IC3 $0.8\mu$ | MCM |
|---|---|---|---|---|
| driver resistance ($\Omega$) | 164 | 212 | 100 | 25 |
| wire resistance ($\Omega/\mu m$) | 0.033 | 0.048 | 0.03 | 0.008 |
| wire capacitance ($fF/\mu m$) | 0.019 | 0.022 | 0.014 | 0.06 |
| sink loading capacitance ($fF$) | 5.7 | 7.06 | 15.3 | 1000 |
| layout area ($mm^2$) | $10^2$ | $10^2$ | $10^2$ | $100^2$ |

Table 1: Technology parameters for three common CMOS IC processes, as well as for a typical MCM process. Parasitics for the IC technologies were provided by MOSIS, while the MCM interconnect parasitics are courtesy of Professor Wayne W.-M. Dai and the AT&T Microelectronics Division.

The SPICE parameters we used are representative of typical MOSIS $0.8\mu$, $1.2\mu$ and $2.0\mu$ CMOS IC processes, as well as a typical MCM technology (see Table 1). Our SPICE delay model uses constant resistance and capacitance values per unit length of interconnect (i.e., both resistance and capacitance are proportional to wirelength). In addition, sink loading capacitances are used at all the pins to model loads driven by the interconnect. In our LDRG implementation, the initial Steiner routing tree was computed using an efficient implementation [2] of the Iterated 1-Steiner algorithm of Kahng and Robins [15], which is known to yield near-optimal Steiner trees [3].

Figure 4 shows the percent improvement in maximum source-sink delay over Steiner routing. Significant improvment is observed for example, in the IC3

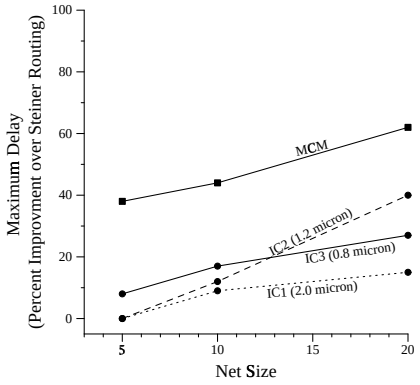


Figure 4: Maximum source-sink delay improvment over Steiner routing. The values shown are percent averages of $(1 - t_{ED}(LRDG) \;/\; t_{ED}(Steiner\ routing))$ over 100 uniformly distributed nets.

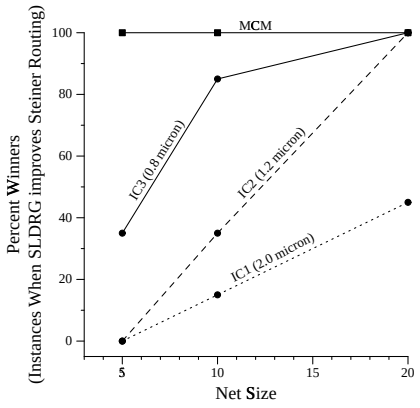


Figure 5: Percent of instances where LDRG improved upon the initial Steiner routing.

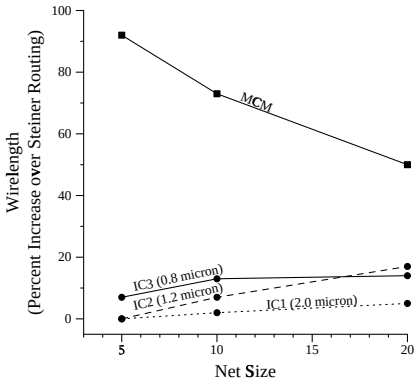


Figure 6: Wirelength increase over Steiner routing. The values shown are percent averages over 100 uniformly distributed nets.

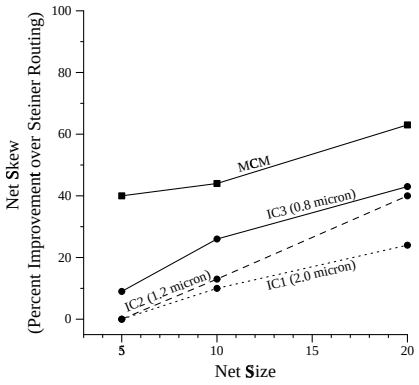


Figure 7: Signal propagation skew improvement over Steiner routing. The values shown are percent averages over 100 uniformly distributed nets.

(0.8$\mu$ CMOS) technology for 10-pin nets, where LDRG wins over Steiner routing by 17% while incurring a 13% wirelength penalty. Even larger improvment is seen in the MCM technology, with improvement of 38% for 5-pin nets and 44% for 10-pin nets.

We tallied the number of cases where LDRG was able to improve upon the initial Steiner routing (see Figure 5). We observe that the number of improvable cases increases with the net size, and approaches 100% for 20-pin nets in all technologies; for MCM routing, LDRG is superior to Steiner routing for all net sizes.

An added benefit of non-tree routing is a significant reduction in signal skew (i.e., the maximum difference between signal arrival time between any two pins). Figure 7 shows the average percent improvment in signal skew over Steiner routing. For example, for 10-pin nets, LDRG yields 44% skew reduction for MCM, 26% skew reduction for IC3, 13% skew reduction for IC2, and 10% skew reduction for IC1.

Most dramatic are the results for nets of 20 pins in the MCM technology, where maximum source-sink delay and skew are both improved by an average of over 60%, while incurring only a 50% extra wirelength penalty. It seems that for our LDRG method the percent improvement in delay is always greater than the percent increase in wirelength for all IC technologies and net sizes. For MCM, the percent increase in wirelength decreases as net size increases.

## 5 Conclusions

In this paper we have explored the consequences of abandoning an implicit restriction common to previous routing formulations, namely the insistence on a strictly acyclic (tree) routing topology. Instead, we reformulated the routing problem as one of constructing a routing *graph* with low maximum source-sink delay. We have shown that adding a few extra wires to an initial routing tree often dramatically improves signal propagation delay by exploiting the tradeoff between the capacitance and resistance in a routing. For example, depending on net size and technology, the addition of several extra wires/edges to an existing Steiner tree routing can improve the average signal propagation delay by up to 62%, while incurring modest wiresize increase.

Our algorithm is efficient, and our basic approaches are amenable to numerous extensions, such as critical sink routing and wiresizing. In addition, non-tree routings offer higher reliability since a single open fault along a cycle in the routing will not disconnect the net; thus, our non-tree routing can tolerate certain short faults due to manufacturing flaws and electromigration. Finally, non-tree routings significantly reduce signal propagation skew and can also mitigate signal reflection [7].

## 6 Acknowledgements

We thank Pak Chan and Kevin Karplus for valuable discussions and for the use of their code in computing Elmore delay for general RC networks.

## References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, *A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing*, in Proc. IEEE Intl. Symp. Circuits and Systems, Chicago, IL, May 1993.

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

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

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

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

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

[7] P. K. Chan, University of California, Santa Cruz, private communication, June 1993.

[8] P. K. Chan and K. Karplus, *Computing Signal Delay in General RC Networks by Tree/Link Partitioning*, IEEE Trans. Computer-Aided Design, 9 (1990), pp. 898–902.

[9] J. Cohoon and J. Randall, *Critical Net Routing*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 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, 11 (1992), pp. 739–752.

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

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

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

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

[15] A. B. Kahng and G. Robins, *A New Class of Iterative Steiner Tree Heuristics With Good Performance*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 893–902.

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

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

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

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

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

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

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