On High-Speed VLSI Interconnects: Analysis and Design*

K. D. Boese, J. Cong, K. S. Leung, A. B. Kahng and D. Zhou†

CS Dept., UCLA, Los Angeles, CA 90024-1596
† EE Dept., UNC Charlotte, Charlotte, NC 28223

Abstract

We survey our recent work in the analysis and design of interconnect topologies for high-speed VLSI. Results include: a new, fast distributed RLC analysis method based on a two-pole approximation; an *A-tree* formulation for performance-driven interconnect; an optimal wiresizing algorithm; and new critical-path dependent routing tree algorithms.

1 Introduction

Interconnection design is becoming a major concern in the design of high-speed systems, where state-of-the-art integrated circuits use submicron technology and operate at multi-giga hertz clock rates. In this range, optimization based on the traditional layout design objective, i.e. minimization of chip area, no longer suffices since the emphasis on system performance requires different consideration. For instance, the minimum Steiner tree has traditionally been the preferred interconnect topology because: (1) it uses the minimum wiring area and (2) minimum wiring area results in minimum wire capacitance, which is the dominant factor in the interconnect delay in the conventional technology. However, in the design of high-speed interconnects, the minimum delay is no longer achieved by a minimum Steiner tree [30]. Figure 1 shows the simulation results of a minimum Steiner tree and a routing tree produced by the BRBC algorithm in [6] (which minimizes both the total wirelength and the longest source-sink path of the routing tree). We can see clearly that the latter tree results in a smaller delay. The performance variation caused by different interconnect structures motivates our study of new interconnect topology for performance optimization. This paper summarizes our ongoing research on the analysis and design of high-speed VLSI interconnects.

In order to study performance-driven interconnect designs, performance analysis is inevitably the first problem to be addressed. Although circuit performance can always be evaluated by a numerical simulator such as SPICE, this is usually very slow and fails to reveal the relationship between interconnect structure and interconnection delay. An analytical closed-form solution is definitely preferred to guide a layout design system. Thus, the first part of this paper treats the analysis of interconnect networks and develops a simple analytical solution for performance evaluation.

The second part of this paper focuses on the construction of interconnect structures for performance optimization. Our study shows convincingly that the interconnection delay is no longer determined only by the total wirelength, but is also strongly affected by the interconnect topology and wire widths at different segments. Efficient algorithms are presented for determining the optimal interconnect topologies and wire width assignments based on distributed RC delay models.

*Partial support was provided by NSF MIP-9110511; by NSF MIP-9110450; by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050; and by a GTE Graduate Fellowship. We are also grateful for the support of Cadence Design Systems under the California MICRO program.

2 Interconnection Performance Analysis

2.1 Existing Works and Motivation

Research on the evaluation of interconnect performance has been pursued in several different directions. The most accurate and complete method of solution is to solve 3-D (or 2-D) *time-variant Maxwell equations* [11, 25]. At this level, the effect of electrical and geometric parameters on circuit performance can be investigated in great detail. For instance, the scattering of waveform at a wire bend (or a discontinuity) can be evaluated precisely [2]. However, due to the extremely high complexity of this approach, it is very difficult to develop a general analytical solution. A simplified approach considers a 1-D problem, i.e., solving a 1-D *telegraph equation* [8]. Even though the dimension of the problem is reduced, only the ideal case, i.e. an infinite long line or ideal termination, has been solved analytically [29]. For general interconnect structures, an analytic solution is almost impossible to obtain because of the irregular boundary conditions encountered in solving the telegraph equation.

Another direction of interconnect performance evaluation involves *circuit simulation* based on numerical methods. SPICE is a commonly used circuit simulator which, in principle, can simulate an arbitrary circuit. However, it has an obvious disadvantage in that a general understanding of physical meaning behind the design is obscured by the numerical calculation.

Therefore, circuit simulation is used to *verify* and *test* a design, but is not able to *directly guide* the design in the first place.

To achieve performance-driven interconnect design, an analytical closed-form solution for the performance evaluation is essential: only an analytical solution can reveal the relationship between the performance and the design parameters, and thus be used as an objective function within the layout design process. Elmore [11] and Rubinstein et al. [25] presented an analytical closed-form solution for estimating the delay in a distributed RC circuit. Chan and Schlag analyzed the lower bound on the delay in an RC-mesh circuit [4]. These works provided an in-depth understanding of the properties of a passive RC network, but their result can only be applied when the effect of inductance is negligible. For *distributed RLC circuits*, Pillage and Rohrer used the so-called AWE method to evaluate circuit performance [20]. In the AWE method, a high order system is approximated by a desired lower order system, resulting in speed-ups of up to a factor of 1,000 over SPICE simulation [23]. However, in applying the AWE method, numerical computation again needs to be involved. This results in the same disadvantage as the circuit simulation that the relationship between the interconnect structure and circuit performance cannot be explicitly explored.

In this section, we present a good 2nd-order approximation method for performance evaluation of a distributed RLC circuit. The 2nd-order approximation is the simplest non-trivial approximation that may characterize the circuit response of an interconnect structure for high speed circuits when inductance is considered. Therefore, it is more likely to afford an analytical solution which can be used to guide the layout design process.

2.2 The Two-Pole Approximation Results

Given a distributed RLC tree with m nodes, assume that the input signal is applied at the root. We want to calculate the voltage response v_k at an arbitrary node k. We use P_k to denote the path from the root to node k, and $z_{n(k)}$ to denote the impedance of the capacitance at node k connected to the ground (called the *node-impedance*). The series resistance and inductance in an edge of the tree is called the *edge-impedance* of the edge, and the *path-impedance* of a path in the tree is defined to be the sum of edge-impedance over all edges in the path. Let $z_{k,j}$ denote the path-impedance of the common portion of the paths P_k and P_j. Let $Z_{k,j}(s)$ and $Z_{n(j)}(s)$ denote the Laplace transform of $z_{k,j}$ and $z_{n(j)}$, respectively. We showed that the system poles can be well approximated by solving

$$\left(\frac{\pi}{2}\right)^2 + \sum_{j=1}^{m} \frac{Z_{k,j}(s)}{Z_{n(j)}(s)} = 0,\ k = 1, \ldots, m. \tag{1}$$

Writing s_j in the form $s_j = -\alpha_j + i\beta_j$ and taking the reverse Laplace transform, we obtain the circuit response at node k

$$v_k(t) = V_0 - \sum_{j=1}^{m} Res(V_k(s_j))e^{(-\alpha_j + i\beta_j)t},\ k = 1, ..., m, \tag{2}$$

where $Res(V_k(s_j))$ is the residue of $V_k(s)$ at pole s_j. By setting the inductance to zero, Eqs.(1) and (2) reduce to those obtained by Rubinstein et al. [25] where a distributed RC-tree is considered. Details of the derivations are presented in [30].

Using Eqs.(1) and (2), we can relate the circuit performance to electrical and geometric parameters such as wire capacitance and wire length. As an example, let us consider a single transmission line of length l driven by a transistor and charging a capacitor as shown in Figure 2. (This is a common interconnection scenario in CMOS circuits.) We uniformly cut the line into m segments and later let $m \to \infty$. Eq.(1) yields the following equation for the system pole calculation if a second order approximation is used.

$$\left(\frac{L_0 C_0 l^2}{2} + L_0 l C_g\right) s^2 + \left(R_d C_0 l + \frac{R_0 C_0 l^2}{2} + (R_0 + R_0 l) C_g\right) s + 1.23 = 0, \tag{3}$$

where C_0, L_0 and R_0 are the capacitance, inductance and resistance per unit length of the wire, and R_d and G_g are the driving transistor's output resistance and loading capacitance, respectively. The relationship between the circuit response at the receiving end and the electrical parameters is obtained by using Eq.(2):

$$V(t) = V_0 - V_0 \left(\frac{s_2}{s_2 - s_1} e^{s_1 t} + \frac{s_1}{s_1 - s_2} e^{s_2 t} \right) \tag{4}$$

where s_1 and s_2 are the solutions of Eq. (3). If we represent R_0, L_0, and C_0 in terms of the width and thickness of the transmission line and substitute them in Eq. (3), we obtain a relation between the circuit response and the geometric parameters. This type of analytical relation is very important to the performance-driven layout design.

We have implemented a two-pole simulator for distributed RLC circuits based on Eqs.(1)-(4).[1] Figures 1 and 2 show the comparison of the circuit responses computed by the two-pole simulator and the SPICE simulator. A fair match is seen consistently for all examples. In order to highlight the necessity of using the distributed RLC-model we also plot in Figure 2 the circuit response based on the distributed RC-model [25] and on the lumped RLC-model [1]. It is clear that neither of them can effectively model the design in the frequency of interest.

3 Performance-Driven Interconnect Design

Limited progress has been reported in the literature for the performance-driven interconnect design problem. In [10], net priorities are determined based on static timing analysis; nets with high priorities are processed earlier using fewer feedthroughs. In [16], a hierarchical approach to timing-driven routing was outlined. In [21], a timing-driven global router based on the A* heuristic search algorithm was proposed in building-block designs. In [6], a timing-driven global router was proposed to minimize both the total wirelength and the longest path from the source to any sink simultaneously. Although these routers tried to reduce the interconnection delay by optimizing the routing topology, their objective functions were oversimplified due to the use of linear delay model or the lumped RC delay model. Moreover, none of these algorithms study the impact of wiresizing on interconnect delay minimization. Although wiresizing was used by Fisher and Kung [12] in H-tree clock routing, the general wiresizing problem for arbitrary routing topology has not been well studied before.

In this section, we study the interconnect design problem under a distributed RC delay model developed by Rubinstein et al. [25]. Given a distributed RC circuit, an upper bound of

[1]Note that Eq. (3) changes for different RLC-trees.

signal delay at any node is computed as follows:

$$t = \sum_{all\ nodes\ k} R_k \cdot C_k \tag{5}$$

where R_k is the resistance between the source and the node k and C_k is capacitance at the node k. The remainder of this section presents the results on the interconnect design and wiresizing problems based on this delay model.

3.1 Interconnect Topology Optimization

Given a routing tree T implementing a net N which consists of a source n_0 and a set of sinks $\{n_1, \ldots, n_m\}$, we shall use Eq. (5) to compute the signal delay $t(T)$ at any node in tree T. In order to accurately model a routing tree as a distributed RC tree, a grid structure is superimposed on the routing plane and each wire segment in the routing tree is divided into a sequence of wires of unit length. Assuming that a unit-grid-length wire has wire resistance R_0 and wire capacitance C_0, Eq. (5) gives an upper bound of the delay of a routing tree:

$$\begin{aligned} t(T) &= \sum_{k \epsilon T} (R_d + R_0 \cdot l_k(T)) \cdot (C_0 + C_k) \\ &= t_1(T) + t_2(T) + t_3(T) + t_4(T) \end{aligned} \tag{6}$$

where

$$t_1(T) = R_d \cdot C_0 \cdot length(T) \tag{7}$$

$$t_2(T) = R_0 \cdot \sum_{all\ sinks\ k} C_k \cdot l_k(T) \tag{8}$$

$$t_3(T) = R_0 \cdot C_0 \cdot \sum_{k \epsilon T} l_k(T) \tag{9}$$

$$t_4(T) = R_d \cdot \sum_{all\ sinks\ k} C_k \tag{10}$$

In these equations, R_d is the driver resistance, $l_k(T)$ is the wirelength from the source to node k in the routing tree T, $length(T)$ is the total wirelength of the routing tree T, and C_k is the *extra* capacitance (besides the wire capacitance) at node k. Notice that the set of nodes includes all grid points in the routing tree T, not just the sinks and the source. For simplicity, we assume that C_k is non-zero only when node k is a sink, but wire capacitance C_0 presents at every node.

We have shown the following results: (i) the term $t_4(T)$ is a always a constant; (ii) the term $t_1(T)$ is minimized when T is a minimum Steiner tree (MST); (iii) the term $t_2(t)$ is minimized when T is a shortest path tree (SPT); and (iv) the term $t_3(T)$ is minimized when T is a quadratic minimum Steiner tree (QMST). These results are discussed in detail in [7]. In what follows, our goal is to design routing tree T such that $t_1(T) + t_2(T) + t_3(T)$ is minimized.

Notice that the relative importance of these terms is determined by the ratio $\frac{R_d}{R_0}$, which we call the *resistance ratio*. Because the resistance ratio was very large in the previous technology, conventional routing techniques focused on total wirelength minimization and used minimum wire width for all segments in order to minimize $t_1(T)$. However, as the technology advances to smaller device dimensions, according to the CMOS scaling rule [1], driver resistance decreases while wire resistance increases, which leads to a significant reduction of the resistance ratio. In this case, the $t_2(T)$ and $t_3(T)$ terms can no longer be ignored in the interconnect design.

Our study shows that the general interconnect topology optimization problem is NP-complete. Moreover, there is no definite correlation between the three terms $t_1(T)$, $t_2(T)$, and $t_3(T)$ for a general interconnect topology. Due to these difficulties, we focus our attention

on a special type of routing topology called the A-tree. A rectilinear Steiner tree T is an *A-tree* if for any node p in T, the path connecting the source n_0 to p in the tree is a shortest path. A-trees are a generalization of the rectilinear Steiner arborescences studied in [22]. There are several reasons that we are interested in A-trees. First, any A-tree is always a shortest path tree (SPT). Therefore, the $t_2(T)$ term is always minimum. Moreover, we can show that a minimum Steiner A-tree (MSA) is also a quadratic minimum Steiner A-tree ($QMSA$) in most cases. Therefore, minimizing $t_1(T)$ is equivalent to minimizing $t_3(T)$. As a result, minimizing total wirelength of an A-tree leads to simultaneous optimization of $t_1(T)$, $t_2(T)$, and $t_3(T)$. Such a harmony is unlikely to be achieved for general routing topologies. Furthermore, our results show that we can develop efficient algorithms for computing A-trees with optimal or near-optimal wirelength. Therefore, we restrict the solutions to the interconnect topology optimization problem to the class of A-trees with a simple cost function (i.e. minimize the total wirelength of the A-tree). The basic approach of our A-tree construction algorithm is to grow a forest of single-node subtrees into an A-tree one step at a time. At each step, we perform an *expanding operation* which either connects two subtrees into a new (larger) subtree or else grows one of the subtrees. Throughout the construction, every subtree in the forest remains an A-tree. We have obtained the following results.

(i) We have developed a set of optimal expanding operations which allows us to grow small A-trees into large A-trees. Given a forest F, let $T_{opt}(F)$ denote an optimal A-tree which contains all the edges in F. An expanding operation is *optimal* if $cost(T_{opt}(F)) = cost(T_{opt}(F'))$, where F is the forest before applying the expanding operation and F' is the resulting forest after applying the expanding operation. We have identified three types of expanding operations and used them extensively in our A-tree constructions.

(ii) We have developed an $O(n^3)$ time heuristic algorithm in which 94% of expanding operations are optimal in practice. As a result, the A-tree constructed by our algorithm is always optimal or near optimal.

(iii) We have developed an efficient procedure which computes a very tight lower bound of the used in an optimal A-tree. The lower bound computation enables us to conclude that 45% of the A-trees constructed by our algorithm are optimal and on average the cost of A-trees by our algorithm are 4% away from the optimal.

Details of these results are presented in [7]. Our experimental results in Section 3.4 show that our A-tree algorithm reduces the interconnection delay by 23% on average as compared to the best known Steiner heuristic.

3.2 Wire Width Optimization

In the preceding subsection, we assume that wires in a routing tree have uniform width, which is widely used in conventional layout designs. However, our study shows that for the high-speed VLSI interconnect designs, proper wiresizing can lead to substantial reduction in signal delay. If the wire width at node k is w_k, according to Eq. (5), the upper bound of the delay of a routing tree becomes:

$$\begin{aligned} t(T) &= \sum_{k \epsilon T} (R_d + R_0 \cdot \sum_{i \epsilon P_k(T)} \frac{1}{w_i}) \cdot (C_0 \cdot w_k + C_k) \\ &= t_1(T) + t_2(T) + t_3(T) + t_4(T) \end{aligned} \tag{11}$$

where

$$t_1(T) = R_d \cdot C_0 \cdot area(T) \tag{12}$$

$$t_2(T) = R_0 \cdot \sum_{all\ sinks\ k} C_k \cdot \sum_{i \epsilon P_k(T)} \frac{1}{w_i} \tag{13}$$

$$t_3(T) = R_0 \cdot C_0 \cdot \sum_{k \epsilon T} \sum_{i \epsilon P_k(T)} \frac{w_k}{w_i} \tag{14}$$

$$t_4(T) = R_d \cdot \sum_{all\ sinks\ k} C_k \quad (15)$$

In these equations, $area(T)$ is the total wiring area of T, and $P_k(T)$ is the path from the source to the node k in the routing tree T. The meanings of the remaining terms are the same as in the previous subsection. Again, we can show that $t_4(T)$ is a constant, and that $t_1(T)$, $t_2(T)$, and $t_3(T)$ have similar physical meanings as in the previous subsection. Given a set of wire segments S implementing a routing tree T and a set of possible widths $W = \{W_1, W_2, \cdots, W_r\}, (W_i < W_{i+1}\ 1 \leq i \leq r-1\}$, the *wiresizing problem* is to find a wire width assignment $f : S \rightarrow W$ such that the delay $t(T)$ defined in Eq. (11) is minimized.

Ideally, we want to determine the routing topology and perform wire width assignment to the segments in the tree at the same time. However, the complexity involved in this problem is very high and the combined problem is unlikely to be solved efficiently. Instead, we adopt the approach of first determining the interconnect topology and then optimizing wire widths in the given routing tree. Our results on the wiresizing problem can be summarized as follows.

(i) We have shown that any optimal wire width assignment is a monotonic assignment. A wire width assignment is *monotonic* if the wire width determined by the assignment is non-increasing along any path from the source to a sink in the tree. Intuitively, the monotonic property suggests that we should use larger width for the segments near the source and smaller width for the segments away from the source for delay optimization.

(ii) We have developed a polynomial-time optimal wire width assignment algorithm for any given routing tree. The time complexity of our algorithm is $O(n^{r-1})$ time, where n is the number of segments in the routing tree and r is the number of wire width choices. Our algorithm is based on a dynamic programming technique, which leads to a significant improvement over the $O(r^n)$ time complexity of the exhaustive enumeration algorithm.

(iii) We have developed a local refinement technique which can be used to compute lower and upper bounds of the optimal width of each wire segment. We say that one wire width assignment is *dominated* by another if the width of any segment specified by the former assignment is no more than the width of the corresponding segment specified by the later assignment. Given a wire width assignment, a *local refinement operation* recomputes the wire width of a particular segment for delay optimization without changing the widths of rest segments. We have proved that such local refinement operations preserve the dominance relationship between the initial assignment and the optimal assignment, i.e., if the initial assignment is dominated by the optimal assignment, the resulting assignment after any number of local refinement operations remains dominated by the optimal assignment, and vice versa. Based on the dominance property of local refinement operations, we developed an $O(n^2 \cdot r)$ time algorithm which computes very tight lower and upper bounds of the optimal width each wire segment. Experimental results show that the lower and upper bounds computed by our algorithm are equal for 99.5% tree segments, and thus immediately yield the optimal wire widths for these segments.

(iv) We have incorporated the lower and upper bound computation into our optimal wire width assignment algorithm. This combined algorithm, called OWSAR (Optimal Wiresizing Algorithm with Refinement), significantly reduced the number of wire width assignments examined by the optimal algorithm. For instance, OWSAR runs 10 times faster than the original optimal wire width assignment algorithm for the case of $n = 16$ and $r = 4$. Hence, it is much more efficient for very large interconnect structures.

Details of these results are presented in [7]. Our experimental results in Section 3.4 show that the OWSAR algorithm further reduces interconnect delay by an additional 21% on average when applied to the A-tree topology.

3.3 Interconnect Topology Design Using Critical Path Information

As performance becomes a dominant system criterion, static timing analysis is iteratively used to determine necessary changes to the performance-driven placement and global routing. In this subsection, we consider the design of routing trees which exploit the critical-path information

that may become available in the course of such a layout synthesis approach.

Existing performance-driven placement methods are essentially of two basic flavors [3]. *Net-dependent* algorithms typically use centroid-connected star cost [26], probabilistic estimates of Steiner tree cost [14], minimum spanning tree cost [9] or the bounding box half-perimeter [19] to estimate wire capacitance and signal delay for a multi-terminal net. Based on this timing analysis, module placements are updated to reduce these "net-based" objective functions for those signal nets which lie along the critical path. *Path-dependent* methods are distinguished by their consideration of delay between the source and a particular *critical sink* of a multi-terminal net. The critical sink is determined via timing analysis using known module delays and estimated path delays. For example, Lin and Du [18] use a linear delay approximation so that their method updates the module placement to reduce the rectilinear distance between sources and critical sinks. Other path-dependent placement methodologies include those due to Hauge et al. [13] and Teig et al. [28].

For any net on a timing-critical path, the path-dependent approach affords an explicit routing constraint with respect to the delay to the net's critical sink. Arguably, net-dependent methods only provide implicit routing constraints, e.g., by limiting the bounding box half-perimeter. However, even with a net-dependent placement methodology it is 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. Given a placement of net N, along with one or more identified critical sinks with possibly varying priorities, our goal is to construct a *critical-path dependent* routing solution $T(N)$ such that the weighted sum of the delays at the critical sinks is minimized. In the following, we consider the special case when exactly one critical sink n_c is specified in the net N.

3.3.1 A Simple Heuristic

The A-tree construction discussed above is not sensitive to the presence of critical sinks. The algorithm creates monotone paths to *all* sinks n_i via its motivation from the *upper bound* on Elmore delay in Eq.(6) – an upper bound that is independent of topology. If we consider the actual Elmore delay formula [11], we may develop useful intuitions for routing tree construction with respect to critical sinks.[2] Figure 3(a)-(c) shows the 1-Steiner tree, the A-tree, and an optimal tree for a given net N with identified critical sink n_c. The 1-Steiner solution has large delay to n_c due to the long source-sink path. On the other hand, monotone paths to every sink in the A-tree result in extra tree capacitance, again implying large delay at n_c. The third construction shows that identification of the critical node clearly affects the optimal routing topology. Moreover, the optimal topology of Figure 3(c) is strongly motivated by the Elmore delay formula: (i) "direct" connection of n_c to n_0 is suggested under certain technological considerations, e.g., for the high-capacitance MCM interconnects an essentially star-like topology will minimize delay at the sinks; and (ii) according to the Elmore model, the number of Steiner points in the n_0 - n_c path should in general be minimized, and/or the Steiner points should be "shifted" toward the source n_0. (In Figure 3(d), note that even though the two trees shown are both A-trees and minimum Steiner trees, the tree on the right has much less signal delay at n_c.)

Finally, note that the optimal tree in Figure 3(c) minimizes total tree cost (as in the 1-Steiner tree), *subject to the path from n_0 to n_c being monotone* (as in the A-tree). This simultaneous optimization of tree radius and tree cost recalls the motivations for the BRBC and A-tree approaches (e.g., Eqs.(6)-(10)), but here we optimize with respect to the critical sink n_c. With this in mind, we produce critical path-dependent interconnection topologies via our *Critical-Sink Routing Tree* (CSRT) algorithm. Given sink locations N and an identified critical sink $n_c \in N$, the CSRT approach is as follows: (1) construct a heuristic (minimum-cost) tree

[2] Let e_v denote the edge from node v to its parent when we root the tree at n_0. The resistance and capacitance of e_v are respectively given by r_{e_v} and c_{e_v}. Let T_v denote the subtree of T rooted at v, and let c_v denote the node capacitance of v. The *tree capacitance* C_v of T_v is the sum of node and edge capacitances in T_v [11] [25]. The Elmore delay is then given by $t_{Elmore}(n_0, n_i) = \sum_{e_v \in path(n_0, n_i)} r_{e_v}(c_{e_v}/2 + C_v)$.

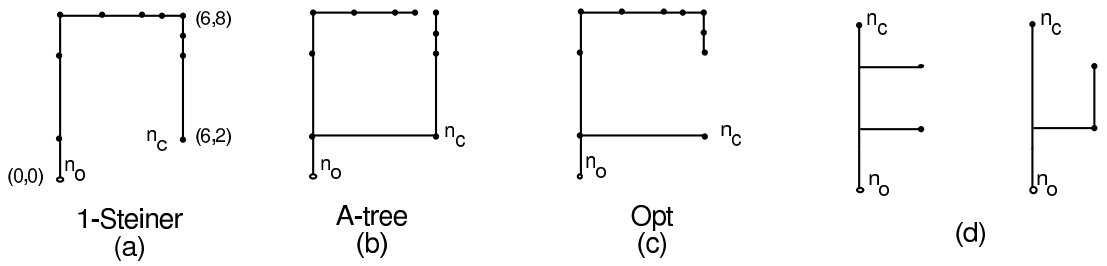


Figure 3: Parts (a)-(c): optimal Steiner tree (cost 2.0cm, delay(n_c) 5.90ns); A-tree (cost 2.5cm, delay(n_c) 4.11ns); and optimal-delay tree (cost 2.2cm, delay(n_c) 3.07ns) for the same sink set. Coordinates are in mm from origin, and IC simulation parameters (Section 3.4) were used. Part (d): two optimal A-trees for a set of three sinks.

T_0 over $N - \{n_c\}$; then (2) make a *direct connection* from n_c to T_0, i.e., use the least possible added wire such that the n_0-n_c path is monotone (see [3] for added details).

Below, we report results for a simple CSRT variant, which we call H1; other variants are discussed in [3]. The H1 method uses the 1-Steiner construction [15] in Step (1); the choice of "direct" connection in Step (2) considers all possible embeddings of the tree edges, i.e., a minimum length *Steiner* connection is made from n_c subject to the monotone path constraint. The complexity of these heuristics is dominated by the complexity of the Step (1) tree construction.[3]

3.3.2 Global Slack Removal

A linear-time postprocessing algorithm, *Global Slack Removal* (GSR), shifts edges in the 1-Steiner output to maximize monotonicity of source-sink paths, without increasing tree cost. If we orient the 1-Steiner tree T by rooting it at the source n_0, a U is defined to consist of three consecutive edges v_1v_2, v_2v_3 and v_3v_4 on a root-leaf path such that the v_1-v_4 pathlength in T is greater than the v_1-v_4 Manhattan distance (Figure 4(a)). The GSR algorithm takes as input a rooted (1-)Steiner tree T and removes U's as shown in Figure 4(b); the input tree is processed in top-down order. We have proved the following results [3]: (i) GSR runs in time linear in the size of T and returns a tree T' which contains no U's; (ii) GSR will return a tree T' such that T and T' have the same total wirelength, (iii) $\forall n_i$, $i = 1, \ldots, |N|$, 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 (iv) the Elmore delay at each n_i in T' is less than or equal to the Elmore delay in T;

3.4 Experimental Results and Discussion

We ran the 1-Steiner (1Stein), A-tree and H1 algorithms on random 4-, 8- and 16-sink inputs. We optionally applied GSA (denoted as +U) to 1-Stein and H1 (note that GSA has no effect on the uniformly monotone A-tree routing); we also applied OWSAR to the A-tree output. Our inputs correspond to two distinct technologies: (i) *IC* is a representative 0.8μ CMOS process, and (ii) *MCM* is typical of current MCM technologies, with lower driver resistance and higher

[3]Using a minimum spanning tree or heuristic minimum Steiner tree in Step (1) is evocative of the BRBC method [6] in the sense of using $\epsilon = 0$ for n_c and $\epsilon = \infty$ for all other sinks; indeed, Cong et al. [6] describe an extension which permits differing ϵ_i values for each sink n_i. However, the $(1 + \epsilon_i) \cdot R$ radius bound is with respect to the net radius R, which is a function of all sink locations; the BRBC extension thus cannot enforce a monotone path to the critical sink. Robins [24] describes how to enforce distinct ϵ_i values with respect to a different R_i value at each sink n_i. Our construction simplifies due to the restriction $\epsilon_i \in \{0, \infty\}$.

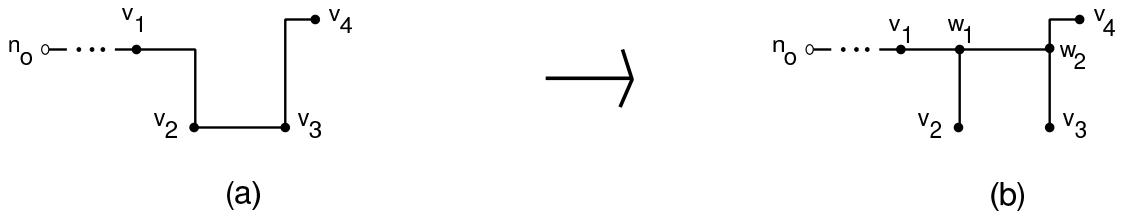


Figure 4: GSR Algorithm: Eliminating source-sink non-monotonicity in a Steiner tree.

wire resistance.[4]

Table 1 gives delay and wire length results and comparisons. The delays at all sink nodes were measured using the two-pole circuit simulator discussed in Section 2. Each entry in the table represents an average taken over every sink node in 50 random point sets. We emphasize that the 1Stein and A-tree algorithms, 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 critical sinks n_i are in some sense "generic". In contrast, the H1 method can return a different tree for each choice of critical sink in the same net. Thus, for the H1 variant we report the delay at n_i in the *specific* H1 tree corresponding to the choice of n_i as the critical sink.

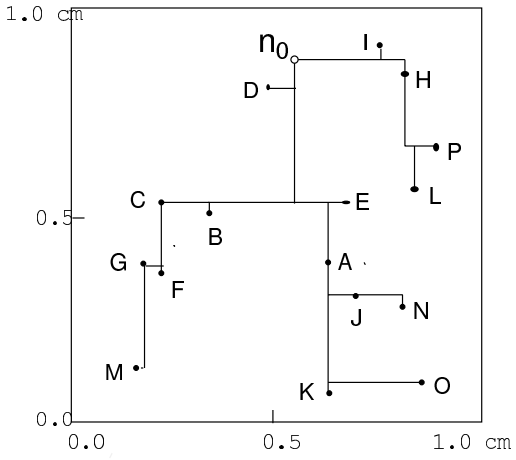
		IC			MCM		
		$\|N\|=4$	$\|N\|=8$	$\|N\|=16$	$\|N\|=4$	$\|N\|=8$	$\|N\|=16$
Ave Delay (ns)	1Stein	2.44	3.48	5.26	10.52	15.18	25.77
	1Stein+U	2.29	3.29	4.87	9.45	14.11	21.95
	H1+U	2.20	3.02	3.93	8.90	11.22	13.81
	A-tree	2.19	2.92	3.89	9.05	11.31	13.73
	Meta	2.17	2.83	3.63	8.84	10.34	11.42
	A+OWSAR	2.05	2.66	3.70	5.72	6.81	8.35
Ave WL (cm)	1Stein	1.51	2.22	3.13	15.65	21.93	31.30
	H1+U	1.58	2.39	3.41	16.20	23.33	33.65
	A-tree	1.53	2.30	3.39	15.71	22.78	34.03
	Meta	1.54	2.33	3.42	16.01	23.38	34.36
Wins (%)	H1+U	8.5	39.3	50.7	21.0	47.8	54.3
	A-tree	18.0	49.0	48.5	6.0	39.8	45.0
Nodewise (H1+U)/(A) Delay Ratio	min	0.97	0.87	0.76	0.87	0.59	0.43
	ave	1.00	1.02	1.00	0.96	0.95	0.93
	max	1.05	1.17	1.27	1.01	1.30	1.81

Table 1: Routing tree results for modern IC and MCM technologies. Notes: (i) all source and sink locations are chosen randomly in layout region with grid resolution $25\mu m$; (ii) H1+U and A-tree wins do not necessarily add up to 100% because of ties; (iii) Min (Max) Nodewise Delay Ratio is geometric average of, for each sink set N, the min (max) H1+U, A-tree delay ratio over all sinks in N; (iv) Meta result for each n_i in each sink set is the tree (A-tree or H1+U tree) which gives smaller delay to n_i; (v) OWSAR results are obtained using only four possible wire widths, $\{1,2,3,4\} \cdot w_0$.

A number of interesting conclusions may be drawn from these results. We see that the A-tree routing is extremely robust: its "generic" delays average out to be slightly *less* than the critical sink-specific delays of the H1+U method, and the total tree weight is competitive even with 1Stein. However, it is clear that for certain choices of critical n_i, the critical-path routing tree can be much better than the generic A-tree routing. For example, in each MCM

[4]Specifics of the technology files (IC,MCM): driver resistance = (100,25) Ω; wire resistance = (0.03,0.008) $\Omega/\mu m$; wire inductance = (492,380) $fH/\mu m$; sink loading capacitance = (15.3,1000) fF; wire capacitance = (0.352,0.06) $fF/\mu m$; layout area = $(10^2, 100^2)$ mm^2.

sink set with $|N| = 16$, one expects to find at least one n_i for which the H1+U delay will be almost 60% less than the A-tree delay. The comparison between H1+U and A-tree shown in Figure 3.4 is very revealing: in general, for critical sinks that are placed close to the source, the H1+U tree can offer a significant delay improvement, but if the critical sink is far away, the A-tree is usually a better choice.[5] With this in mind, we also report the percentage of "wins" for each algorithm on a node-by-node basis. The existence of distinct "domains of expertise" for the A-tree and CPRT approaches suggests that the meta-heuristic [15] ("Meta" in Table 1) which simply chooses the better of the two solutions will be successful; this indeed seems to be the case. If multiple critical sinks are specified for a single net, the A-tree approach seems promising, as it is more "net-oriented". Finally, we observe that the optimal wiresizing method OWSAR gives very significant improvements in source-sink delays: it saves an *average* of 23% percent of delay when applied to the A-tree routing, even when only four distinct wire widths are allowed. Note that OWSAR is also easily applied to trees routed based on critical path information (e.g., H1 trees).



Sink	(H1+U)/(A) Delay		Ratio
A	3.39/4.49	=	0.679
B	3.34/4.88	=	0.684
C	3.63/5.07	=	0.716
D	2.31/2.87	=	0.805
E	3.70/4.57	=	0.810
F	4.44/5.28	=	0.841
G	4.51/5.31	=	0.849
H	2.54/2.88	=	0.882
I	2.53/2.77	=	0.913
J	4.84/5.19	=	0.932
K	5.27/5.46	=	0.965
L	3.08/3.11	=	0.990
M	5.84/5.50	=	1.062
N	7.88/5.24	=	1.122
O	7.69/5.61	=	1.371
P	5.40/3.10	=	1.742

Figure 5: Random 16-sink IC example (1cm by 1cm layout region), showing nodewise ratios of H1+U delays to A-tree delays. The A-tree solution is shown in the figure.

4 Future Work

Our current research in the area of interconnect analysis is aimed at introducing a proper delay definition and developing a closed-form expression for the delay calculation of an under-damped system, where the non-monotonic response presents a great difficulty. In the area of interconnect design, we are extending our work in several directions, including investigation of the interconnect topology design and wiresizing problems under distributed RLC models, consideration of the cross-talk effect between different routing trees, and extensions of critical path-dependent routing tree design to the general case of multiple critical sinks with prescribed delay bounds.

References

[1] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990, pp. 81-133.

[5] Recall the intuition above, namely that critical-path routing will be useful if the technology favors star-like, "direct" connections to the n_i. Note also that timing-driven placement algorithms may tend to place the critical sink n_c close to the source.

[2] L.V. Blake, *Transmission Lines and Waveguides*, Wiley, New York, 1969.
[3] K. D. Boese and A. B. Kahng, "Routing Tree Constructions for Critical Path Optimization", *technical report* UCLA CSD TR-920039, 1992.
[4] P. K. Chan and M. D. F. Schlag, "Bounds on Signal Delay in RC Mesh Networks", *IEEE Trans. on CAD* 8(6) (1989), pp. 581-588.
[5] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global Routing Based on Steiner Min-max Trees", *IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 2-5.
[6] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably Good Performance-Driven Global Routing", *IEEE Trans. on CAD* 11(6), June 1992, pp. 739-752.
[7] J. Cong, K. S. Leung and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model", *UCLA CSD Tech. Report*, Sept. 1992.
[8] A. R. Djordievic, T.K. Sarkar and R. F. Harrington, "Analysis of Lossy Transmission Lines with Arbitrary Nonlinear Terminal Networks", *IEEE Trans. on Microwave Theory and Techniques* 34(6) (1986), pp. 660-666.
[9] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy and R. I. McMillan, "Timing Driven Placement Using Complete Path Delays", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 84-89.
[10] A. E. Dunlop, V. D. Agrawal, D. N. Deutsh, M. F. Jukl, P. Kozak and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting", *Proc. ACM/IEEE Design Automation Conf.*, 1984, pp. 133-136.
[11] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers", *J. Applied Physics* 19 (1948), pp. 55-63.
[12] A. L. Fisher and H. T. Kung, "Synchronizing Large Systolic Arrays", *Proc. SPIE* 341, May 1982, pp. 44-52.
[13] P. S. Hauge, R. Nair and E. J. Yoffa, "Circuit Placement for Predictable Performance", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1987, pp. 88-91.
[14] M. A. B. Jackson and E. S. Kuh, "Estimating and Optimizing RC Interconnect Delay During Physical Design", *Proc. IEEE Intl. Conf. on Circuits and Systems*, 1990, pp. 869-871.
[15] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance", *IEEE Transactions on CAD* 11(7), July 1992, pp. 893-902.
[16] E. Kuh, M. A. B. Jackson and M. Marek-Sadowska, "Timing-Driven Routing for Building Block Layout", *Proc. IEEE International Symposium on Circuits and Systems*, pp. 518-519, 1987.
[17] K. W. Lee and C. Sechen, "A New Global Router for Row-Based Layout", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1988, pp. 180-183.
[18] I. Lin and D. H. C. Du, "Performance-Driven Constructive Placement", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 103-106.
[19] M. Marek-Sadowska and S. Lin, "Timing Driven Placement", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 94-97.
[20] L.T. Pillage and R.A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis", *IEEE Trans. on CAD* 9(4) (1990), pp. 352-366.
[21] S. Prastjutrakul and W. J. Kubitz, "A Timing-Driven Global Router for Custom Chip Design", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 48-51.
[22] S. K. Rao, P. Sadayappan, F. K. Hwang and P. W. Shor, "The Rectilinear Steiner Arborescence Problem", *Algorithmica* 7 (1992), pp. 277-288.
[23] C.L. Ratzlaff and N. Gopal and L.T. Pillage, "RICE: Rapid Interconnect Circuit Evaluator", *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 555-560.
[24] G. Robins, "On Optimal Interconnections", Ph.D. Thesis, CS Dept., University of California, Los Angeles, June 1992.
[25] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. on CAD* 2(3) (1983), pp. 202-211.
[26] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm for Small-Cell ICs", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 48-51.
[27] S. Sutanthavibul and E. Shragowitz, "Adaptive Timing-Driven Layout for High Speed VLSI", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 90-95.
[28] S. Teig, R. L. Smith and J. Seaton, "Timing Driven Layout of Cell-Based ICs", *VLSI Systems Design*, May 1986, pp. 63-73.
[29] D. Zhou, F. P. Preparata and S. M. Kang, "Interconnection Delay in Very High-speed VLSI", *IEEE Trans. on Circuits and Systems* 38(7), 1991.
[30] D. Zhou, S. Su, F. Tsui, D. S. Gao and J. Cong, "Analysis of Trees of Transmission Lines", *technical report* UCLA CSD-920010.