

Performance-Driven Interconnect Design Based on Distributed RC Delay Model

Jason Cong and Kwok-Shing Leung
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024

Dian Zhou
Department of Electrical Engineering
University of North Carolina
Charlotte, NC 28213

Abstract

In this paper, we study the interconnect design problem under a distributed RC delay model. We study the impact of technology factors on the interconnect designs and present general formulations of the interconnect topology design and wiresizing problems. We show that interconnect topology optimization can be achieved by computing optimal generalized rectilinear Steiner arborescences and we present an efficient algorithm which yields optimal or near-optimal solutions. We reveal several important properties of optimal wire width assignments and present a polynomial time optimal wiresizing algorithm. Extensive experimental results indicate that our approach significantly outperforms other routing methods for high-performance IC and MCM designs. Our interconnect designs reduce the interconnection delays by up to 66% as compared to those by the best known Steiner tree algorithm.

1 Introduction

As the VLSI fabrication technology reaches submicron device dimension and gigahertz frequency, interconnection delay has become the dominant factor in determining circuit speed [1]. Most previous works on the interconnect design problem are based on a simplistic linear delay model of wirelength minimization, and many global routing algorithms based on the Steiner tree formulation have been proposed, such as [2, 11]. Although these methods produce good results in term of wirelength minimization, they cannot achieve performance optimization in the design of high-speed ICs and MCMs. Moreover, uniform wire width has been used for the connections of most signal nets in the conventional routing methods. Very little is known about proper wiresizing for delay optimization. As a result, there is a strong need for systematic studies in the interconnect topology design and wiresizing problems for delay optimization in high-performance system designs.

However, limited progress has been reported in the literature for the performance-driven interconnect design problem. In [6], net priorities are determined based on static timing analysis; nets with high priorities are processed earlier using fewer feedthroughs. In [10], a hierarchical approach to timing-driven routing was outlined. In [12], a timing-driven global router based on the A* heuristic search algorithm was proposed in building-block designs. In [3], 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 studies the impact of wiresizing on interconnect delay minimization. Although wiresizing was used by Fisher and Kung [8] in H-tree clock routing, the general wiresizing problem for arbitrary routing topology has not been well studied before.

In this paper, we study the interconnect design problem under a distributed RC delay model developed by Rubinstein, Penfield and Horowitz [14]. Using this model, we develop a routing algorithm based on the efficient construction of rectilinear arborescences. We have also developed a polynomial time optimal wiresizing algorithm which is applicable to arbitrary routing topology.

2 Problem Formulation

In this paper, we use a distributed RC delay model developed by Rubinstein, Penfield, and Horowitz [14]. Given a distributed RC circuit, the signal delay at a node is computed as follows:

$$t = \sum_{\text{all nodes } k} R_k^* \cdot C_k^* \quad (1)$$

where R_k^* is the resistance between the source and the node k and C_k^* is capacitance at the node k .

There are several reasons that we choose this delay model: (i) Although this model is simpler than the commonly used Elmore delay model [7] for distributed RC circuits, it still captures the distributed nature of the circuit; (ii) It gives an accurate upper bound of signal delay at every node and correlates very well with the Elmore delay [14]; (iii) It is much easier to use for interconnect design optimization since it gives a uniform upper bound of the delay at every node.

Given a routing tree T implementing a net which consists of a source and a set of sinks, we shall use (1) to compute the signal delay $t(T)$ at any node in tree T . In order to model a routing tree as a distributed RC tree accurately, 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 as shown in Figure 1. (Adjacent grid points are unit length apart.) When the wire width is fixed, both the wire resistance and the wire capacitance are proportional to the wirelength.

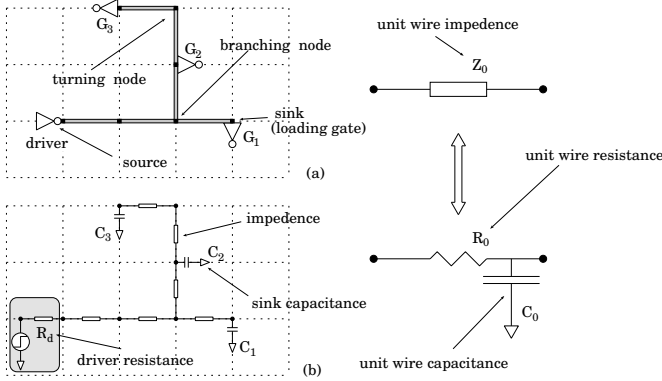


Figure 1: A grid structure for the distributed RC delay model. (a) The layout of an interconnect T with 3 sinks (G_1 , G_2 , and G_3). (b) The corresponding distributed RC model of T . Each edge in T connecting two adjacent nodes is modeled as a RC element containing a resistance R_0 and a capacitance C_0 . Each sink has an extra loading capacitance (C_1 , C_2 , and C_3 in this example).

(We shall discuss the case of variable wire widths later.) Assume that a unit-grid-length wire has wire resistance R_0 and wire capacitance C_0 , then according to (1), an upper bound of the delay of a routing tree is:

$$\begin{aligned}
 t(T) &= \sum_{k \in T} (R_d + R_0 \cdot |P_k(T)|) \cdot (C_0 + C_k) \\
 &= \sum_{k \in T} R_d \cdot C_0 + \sum_{k \in T} R_0 \cdot |P_k(T)| \cdot C_k + \\
 &\quad \sum_{k \in T} R_0 \cdot |P_k(T)| \cdot C_0 + \sum_{k \in T} R_d \cdot C_k \quad (2)
 \end{aligned}$$

where R_d is the driver resistance, $P_k(T)$ is the path from the source to node k in the routing tree T , $|P_k(T)|$ is the length of the path $P_k(T)$, and C_k is the *extra* capacitance (besides the wire capacitance) at node k in T . Notice that the set of nodes includes all grid points in the routing tree T , not just sinks and branching nodes. If node k is a sink, then C_k is the loading capacitance at the node. If node k is a via, it can also be formulated by adding some extra capacitance at the node [16], which reflects the distributed nature of interconnection delay. For simplicity, we assume that C_k is non-zero only when node k is a sink, but wire capacitance C_0 is present at every node. Note that the summation in (2) is over all nodes in the routing tree T , not just sinks. Based on this delay model, we shall formulate a number of performance optimization problems in the subsequent subsections.

2.1 Optimal Interconnect Topology Design

We can rewrite (2) as follows:

$$t(T) = t_1(T) + t_2(T) + t_3(T) + t_4(T) \quad (3)$$

where

$$t_1(T) = R_d \cdot C_0 \cdot \text{length}(T) \quad (4)$$

$$t_2(T) = R_0 \cdot \sum_{\text{all sinks } k} C_k \cdot |P_k(T)| \quad (5)$$

$$t_3(T) = R_0 \cdot C_0 \cdot \sum_{k \in T} |P_k(T)| \quad (6)$$

$$t_4(T) = R_d \cdot \sum_{\text{all sinks } k} C_k \quad (7)$$

In (4)-(7), $\text{length}(T)$ is the total wirelength of the routing tree T , and $|P_k(T)|$ is the path length from the source to the node k . Since the driver resistance R_d and the total loading capacitance at all sinks, $\sum_{\text{all sinks } k} C_k$, are fixed for a given net, $t_4(T)$ is a constant. In order to minimize $t(T)$, we only need to minimize $t_1(T) + t_2(T) + t_3(T)$. It is interesting to see the physical meanings of these three terms.

The first term $t_1(T)$ is the product of the driver's output resistance and the total wire capacitance. This term is minimized when $\text{length}(T)$ is minimum. In the conventional technology, $t_1(T)$ is the dominating term since the driver's output resistance is much larger than the wire resistance¹ (i.e. $R_d \gg R_0 \cdot \text{length}(T)$). This explains partially why the conventional routing methods emphasize minimization of the total wirelength. Clearly, minimizing $t_1(T)$ leads to an optimal Steiner tree (*OST*).

In the summation of the second term $t_2(T)$, each term is a product of the loading capacitance C_k of each sink and the pathlength $|P_k(T)|$ from the source to the sink k in T . Since the loading capacitance of a sink in a given design is fixed, $t_2(T)$ is minimized when all paths from the source to sinks are minimum in length, which results in a shortest path tree rooted at the source. This shows that when all (or some) loading capacitances are very large, we need to select the shortest paths to connect the source to these sinks. Therefore, minimizing $t_2(T)$ leads to a shortest path tree (*SPT*).

The third term is more interesting. It is proportional to the summation of the pathlengths from the source to all nodes (not just sinks) in T . Intuitively, if there is a long source-to-leaf path P in T , $t_3(T)$ will be very large since it contains a term $\sum_{k \in P} |P_k(T)|$ which is roughly proportional to the square of the length of P . Therefore, we prefer a routing tree of short paths in order to keep $t_3(T)$ small. This, in fact, justifies the work by Cong, et al. [3] in which the radius (i.e. the longest source-sink path) of a routing tree is kept under certain bound in the layout design. On the other hand, if $\text{length}(T)$ is large, the number of nodes in T is large and it may increase $t_3(T)$ as well. Therefore minimizing $t_3(T)$ leads to a routing tree which is "in-between" a shortest path tree and an optimal Steiner tree. We shall call a tree optimal under the cost function $t_3(T)$ a *quadratic minimum Steiner tree (QMST)*.

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, $t_2(T)$ and $t_3(T)$ can no longer be ignored in the design of next-generation VLSI circuits. Although Equations (4)-(7) are obtained from a distributed RC

¹In the $2\mu\text{m}$ CMOS technology, the typical values of driver resistance and unit wire capacitance are: $R_d = 1\text{k}\Omega \sim 4\text{k}\Omega$, $R_0 = 0.03\Omega/\mu\text{m}$.

delay model, the impact of the resistance ratio on interconnection design is true for distributed RLC delay models as well. Zhou et al. studied the interconnection delay in a single high-speed transmission line and have observed similar results [15].

In general, the *minimum delay routing tree (MDRT) problem* can be formulated as follows: Given a signal net $N = \{N_0, N_1, \dots, N_{n-1}\}$ where N_0 is the source and $N - \{N_0\}$ is the set of sinks, and three nonnegative constants α , β , and γ , find a rectilinear routing tree T of the net N such that

$$\alpha \cdot \text{length}(T) + \beta \cdot \sum_{\text{all sinks } k \in N} |P_k(T)| + \gamma \cdot \sum_{\text{all nodes } k \in T} |P_k(T)| \quad (8)$$

is minimized, where $\text{length}(T)$ is the total wirelength of the tree T and $|P_k(T)|$ is the pathlength from the source N_0 to the node k in T . Note that the MDRT problem is NP-hard in general since the problem degenerates into the classic Steiner tree problem when both β and γ equal zero.

2.2 Optimal Wire sizing

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 new generation of VLSI technology, proper sizing of wires can lead to substantial reduction in signal delay.

Assume that we have a set of discrete wire widths, $\{W_1, W_2, \dots, W_r\}$, to choose for each wire segment. Let w_k be the width of the incoming grid edge $e(k)$ at node k . According to (1), the upper bound of the delay of a routing tree becomes:

$$t(T) = t_1(T) + t_2(T) + t_3(T) + t_4(T) \quad (9)$$

where

$$t_1(T) = R_d \cdot C_0 \cdot \text{area}(T) \quad (10)$$

$$t_2(T) = R_0 \cdot \sum_{\text{all sinks } k} C_k \cdot \sum_{i \in P_k(T)} \frac{1}{w_i} \quad (11)$$

$$t_3(T) = R_0 \cdot C_0 \cdot \sum_{k \in T} \sum_{i \in P_k(T)} \frac{w_k}{w_i} \quad (12)$$

$$t_4(T) = R_d \cdot \sum_{\text{all sinks } k} C_k \quad (13)$$

In these equations, $\text{area}(T)$ 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 . Again, we can show that $t_4(T)$ is a constant, and $t_1(T)$, $t_2(T)$, and $t_3(T)$ have similar physical meanings as in the previous subsection.

The wire sizing problem can be formulated as follows: Given a set of wire segments S implementing a routing tree T and a set of possible widths $W = \{W_1, W_2, \dots, W_r\}$, the wire sizing problem is to find a wire width assignment $f : S \rightarrow W$ such that the delay $t(T)$ defined in (9) is minimized.

We assume that each wire segment (rather than each grid edge) has a set of discrete choices of wire widths since this resembles more closely to the realistic design style and reflects the actual technological constraint where arbitrary width variation within a segment is usually undesirable².

²Nevertheless, this segment-based formulation can easily be generalized to handle the case where variable wire width is allowed within a segment by introducing artificial nodes along each segment.

Ideally, we want to determine the topology of the routing tree and the wire width assignment of the segments in the tree at the same time. However, the complexity involved in this problem is very high and it 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. Solutions to the interconnect topology design and wire sizing problems will be discussed in Section 3 and 4 respectively. Due to the length restriction, we omit all proofs of the theorems presented in the next two sections. The reader may refer to [4] for details.

3 Interconnect Topology Design

Since the general MDRT problem is NP-hard and there is no definite correlation between the three terms $t_1(T)$, $t_2(T)$, and $t_3(T)$ for a general interconnect topology, we shall focus our attention on a special type of routing topology, called the A-tree topology, which is defined as follows.

Definition 1 A rectilinear Steiner tree T is called an A-tree if every path connecting the source N_0 and any node p on the tree is a shortest path.

Notice that A-trees are generalization of the rectilinear Steiner arborescences studied in [13]. Given a set N of n nodes lying in the first quadrant of E^2 (the Euclidean Plane), including a node at the origin, a *rectilinear Steiner arborescence* is a directed tree rooted at the origin and contains all nodes in N , composed of horizontal and vertical arcs oriented only from left to right or from bottom to top. An A-tree, however, allows the nodes in N to lie anywhere in E^2 as long as paths are always directed away from the origin.

There are several reasons that we are interested in A-trees. First, any A-tree is always a SPT. Therefore, the term $t_2(T)$ is always minimum. Moreover, we show in [4] that an optimal Steiner A-tree (OSA) 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 $t_3(T)$ (subject to the A-tree topology). Such a harmony would be impossible to achieve for general routing topologies. Furthermore, we can develop efficient algorithm for computing A-tree with optimal or near-optimal wirelength. Therefore, we have very good reasons to restrict the solutions to the MDRT problem to the class of A-trees.

The outline of the A-tree algorithm is as follows: Starting with a forest of n single-node arborescences, we apply a sequence of moves. Each move introduces a path consisting of one or more segments, which either “grows” an existing arborescence, or “combines” two arborescences into a new arborescence. This process is completed when there is only one arborescence left in the forest (see Figure 2). The remaining question is how to choose a good move.

Definition 2 Let F_k be the forest constructed after the k^{th} move by the A-tree algorithm. We define $T^*(F_k)$ as the minimum-cost rectilinear Steiner arborescence containing F_k as a subgraph.

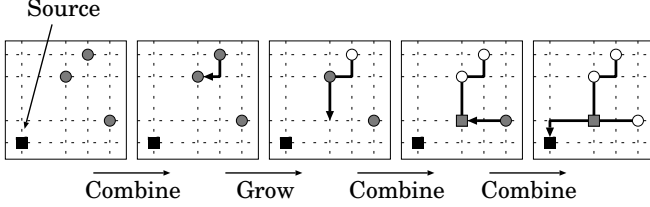


Figure 2: Illustration of how an arborescence is constructed.

According to this definition, $T^*(F_0)$ is the optimal rectilinear Steiner arborescence. $T^*(F_1)$ is the optimal Steiner arborescence containing the path generated by the first move, and $T^*(F_M)$ is the arborescence constructed by the A-tree algorithm, where M is the total number of moves generated by the A-tree algorithm. Let $cost(T)$ be the total wirelength of T . Clearly, since $F_k \subset F_{k+1}$, $cost(T^*(F_k)) \leq cost(T^*(F_{k+1}))$.

Definition 3 If $cost(T^*(F_{k+1})) = cost(T^*(F_k))$, the $k + 1^{th}$ move is called an *optimal move*.

In our algorithm, we introduce three types of moves called *safe moves*: The first type of move *combines* two arborescences into a new one, and the second and third types of moves *grow* an existing arborescence. Detailed descriptions of the safe moves are in [4]. We have shown that all three types of safe moves are optimal moves. Therefore, an A-tree constructed using the safe moves is an optimal OSA. Moreover, we have shown that an A-tree constructed using the safe moves is also an optimal QMSA.

However, it is possible that after a sequence of safe moves, no safe move exists with respect to the current forest. In this case, we need to perform a *heuristic move*, a move that may not be optimal. Two types of heuristic moves are used in our algorithm, and both of them heuristically *combine* two arborescences into a new one. The reader may refer to [4] for details of the complete A-tree algorithm.

Our experimental results indicate that in practice 94% of the moves used by our A-tree algorithm are optimal, and 45% of the A-trees constructed by our algorithm are optimal under both of the OST and QMST cost function because they are generated by safe moves only. Moreover, we have developed a constructive method to compute a lower bound of the wirelength of the optimal Steiner arborescence using the information obtained during the A-tree construction. Based on this computation, we show that arborescences generated by our A-tree algorithm are on average at most 4% from the optimal.

4 Wiresizing

Given a fixed tree T and an assignment f , we denote w_i as the width assignment of segment S_i , and we denote f^* as the optimal wire width assignment. We use $ans(S_i)$ to denote the set of all segments on the unique path from the source N_0 to the segment S_i , excluding S_i , and $des(S_i)$ to denote the set of segments $\{S_k \mid S_i \in ans(S_k)\}$. Moreover, we use $T_{SS}(S_i)$ to denote the single-stem subtree consisting of S_i (as the stem) and the set $des(S_i)$, and $sink(S_i)$ to refer to the set of all sinks on $T_{SS}(S_i)$.

First, we can show that the wiresizing problem has the following properties:

Theorem 1 (Monotone property) Given a routing tree T and an optimal wire width assignment f^* on T , $w_p^* \geq w_c^*$ whenever segment S_p is an ancestor of segment S_c .

Theorem 2 (Separability) The width assignment of segment S_i depends only on the widths of its ancestors and its descendants.

As a result, once S_i and every segment in $ans(S_i)$ are assigned the appropriate widths, the optimal wire width assignment for the single-stem subtrees $T_{SS}(S_{i,1})$, $T_{SS}(S_{i,2})$, \dots , $T_{SS}(S_{i,k})$ (with respect to the width assignment of S_i and segments in $ans(S_i)$) can be *independently* determined, where the segments $S_{i,1}, \dots, S_{i,k}$ are the children of S_i .

Assume we are given a single-stem tree $T_{SS}(S_i)$ with stem segment S_i , and a set of possible widths $\{W_1, W_2, \dots, W_r\}$, we can determine the optimal assignment f^* on $T_{SS}(S_i)$ by enumerating all the possible width assignments of S_i . For each of the possible width assignment W_k of S_i ($1 \leq k \leq r$), we determine the optimal assignment for each single-stem subtree $T_{SS}(S_{i,j})$ independently by recursively applying the same procedure to each $T_{SS}(S_{i,j})$ with $\{W_1, W_2, \dots, W_k\}$ as the set of possible widths (to guarantee the monotone property). The optimal assignment for S_i is the one which gives the smallest total delay. This algorithm is called the *optimal wiresizing algorithm (OWSA)*³. It is not difficult to show that the OWSA algorithm has a worst case time complexity of $O(|S|^{r-1})$, where $|S|$ is the number of segments. Note that our optimal wiresizing algorithm is a significant improvement over the brute-force enumeration method which has complexity $O(r^{|S|})$. Nevertheless, we would like to further improve the runtime of the OWSA algorithm.

Given a routing tree T , a wire width assignment f on S , and a particular segment $S_i \in T$, a *local refinement* on S_i is defined as the operation to determine the optimal segment width of S_i (with respect to the fixed assignment of f on the other segments). \tilde{w}_i is called the locally optimal width of S_i with respect to f . Based on this operation, we have developed a greedy algorithm: Starting with an initial wire width assignment (say, all segments have the minimum width), we traverse the tree and perform refinement on each segment whenever possible. This process is repeated until no improvement is achieved on any segment in the last round of traversal. This is called the *Greedy Wiresizing Algorithm (GREWSA)*. It would not be difficult to show that the complexity of GREWSA is $\Omega(|S|^2 \cdot r)$. Despite its greedy nature, GREWSA performs very well in terms of the quality of assignments and runtime (see Section 5). In fact, GREWSA generates optimal assignments when there are only two choices of wire widths [4].

Given two wire width assignments f and f' on the same tree T , we say $f \succeq f'$ if $w(f, S_i) \geq w(f', S_i)$ for all $S_i \in T$. We can show that assignments generated by GREWSA have the *dominance property*:

³If the original routing tree T is not a single-stem tree, we can decompose T into b single-stem trees, where b is the degree of the root of T , and apply the algorithm to each individual single-stem tree separately.

Theorem 3 (Dominance Property) *Given a wire width assignment f (possibly suboptimal) and a segment S_i on the routing tree T . Assume that f' is the assignment obtained from f by performing a local refinement on the wire segment S_i . Then, $f' \succeq f^*$ ($f' \preceq f^*$) if and only if $f \succeq f^*$ ($f \preceq f^*$), where f^* is the optimal assignment.*

Theorem 3 immediately suggests the strategy of using GREWSA to compute the lower and upper bounds of each segment width of the optimal assignment. If we start with the initial assignment where each segment has the minimum wire width, the resulting assignment computed by GREWSA gives a lower bound of the optimal width for each segment, since each intermediate assignment computed by GREWSA, including the last one, is dominated by the optimal assignment. Similarly, if we start with an initial assignment where each segment has the maximum wire width, the resulting assignment computed by GREWSA gives an upper bound of the optimal width for each segment.

We can further combine OWSA and GREWSA as follows: First, we obtain the lower and upper bounds of each wire segment using the GREWSA algorithm. Then, we run a modified version of OWSA which only considers assignments whose segment widths satisfy the lower and upper bounds computed by the GREWSA algorithm. Since the lower and upper bounds obtained from the GREWSA algorithm are very close or even identical in most cases, the total number of candidate assignments ever generated by OWSA algorithm is much smaller than that by the OWSA algorithm alone. As a result, the upper and lower bounds obtained help speed-up the algorithm significantly. For instance, the combined GREWSA-OWSA algorithm runs 10 times faster than OWSA for the case of $|S| = 16$ and $r = 4$.

5 Experimental Result

We have implemented the A-tree algorithm and the wire-sizing algorithms (GREWSA and OWSA) in ANSI C for the IBM-PC and Sun SPARC station environments. We have compared the A-tree and wire-sizing algorithms with other existing routing techniques on both MCM and advanced IC technologies. Section 5.1 shows the improvement achieved by the A-tree and wire-sizing algorithms over existing routing algorithms, and Section 5.2 shows the impact of resistance ratio and transistor sizing.

5.1 Effect of Interconnect Topology Optimization and Wire-sizing

The results in this section are based on a typical MCM technology⁴ [5]. We tested our algorithms on signal nets of 4, 8, and 16 sinks. For each net size, 100 nets were generated on a 100 mm x 100 mm routing region for the MCM technology. The grid resolution is 25 μm per unit-grid-length.

We have compared the generalized A-tree algorithm with the 1-Steiner algorithm proposed by Kahng and Robins [9], and the bounded-radius-bounded-cost (BRBC) algorithm proposed by

⁴Specifics of the MCM technology file: driver resistance = 25 Ω ; wire resistance = 0.008 $\Omega/\mu\text{m}$; loading capacitance = 1000 fF; wire capacitance = 0.060 fF/ μm ; wire inductance = 380 fH/ μm ; total area = 100 mm x 100 mm.

Comparison	Algorithm	4 sinks	8 sinks	16 sinks
Wirelength (mm)	1-Steiner	149.55	218.98	310.50
	BRBC-0.5	182.30	286.18	421.50
	BRBC-1.0	167.48	261.03	385.00
	A-tree	150.98	229.35	339.00
Delay (ns)	1-Steiner	9.10	14.57	26.14
	BRBC-0.5	8.09	11.85	21.04
	BRBC-1.0	7.88	12.57	23.31
	A-tree	8.07	10.49	14.92
	A-tree + GREWSA	5.27	6.57	8.94

Table 1: The average wirelength (in millimeters) and delay (in nanoseconds) are compared. BRBC-0.5 and BRBC-1.0 are two parameterized versions of the BRBC algorithm with the control parameters ϵ chosen to be 0.5 and 1.0, respectively. In all cases the runtime of the A-tree algorithm is less than 0.3 seconds.

Cong et al. [3]. The average delay and wirelength are compared in Table 1. The average delay of each net is obtained by averaging the signal delay at every sink using the two-pole circuit simulator developed by Zhou et al. [16]. Extensive experimental results have shown that the two-pole simulator is comparable to SPICE in terms of delay simulation, but runs much faster [16]. In fact, the two-pole simulator computes the Elmore delay when the given circuit is a RC circuit.

The 1-Steiner algorithm is one of the best known Steiner heuristics [9] and it is not surprising that it has the best wirelength. However, the experimental results show that wirelength minimization does not necessarily lead to minimum delay. In fact, the reduction in the SPT and QMST cost by the A-tree algorithm offsets the wirelength advantage of the 1-Steiner algorithm. Moreover, the performance advantage of the A-tree algorithm over the 1-Steiner algorithm becomes more significant as the net size becomes larger. This is because the distributed nature of interconnect structures become more significant for large nets. As a result, the A-tree algorithm reduces the average by up to 43% as compared to the 1-Steiner algorithm.

The results also show that the optimal wire-sizing can further reduce average delay in routing trees by up to 40%. Moreover, the wire assignment solutions generated by GREWSA are very close to optimal in practice. In more than 95% of the cases, the lower and upper bounds of the optimal wire widths computed by GREWSA uniquely determine the optimal assignments. Therefore, GREWSA-OWSA generates the optimal wire width assignments with far less computation time. In general, the runtime of OWSA is very sensitive to the parameter r , but the runtime of GREWSA-OWSA hardly changes as r increases. For instance, it takes 0.538 seconds for OWSA and 0.062 seconds for GREWSA-OWSA to generate the optimal assignment for a 16-sink net when $r = 4$, and 4.710 seconds and 0.060 seconds respectively when $r = 6$.

5.2 Effect of the Resistance Ratio and Transistor Sizing

We have studied the impact of the resistance ratio and transistor sizing on the improvement of the A-tree algorithm over the 1-Steiner algorithm. In particular, we have compared the

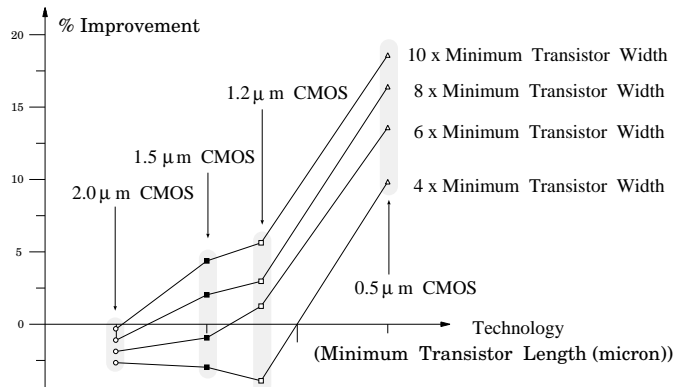


Figure 3: Performance improvement of the A-tree algorithm over the 1-Steiner algorithm in terms of the average delay as a function of the IC technology and the transistor size.

algorithms under four IC technologies, including the 2.0 μm , 1.5 μm , 1.2 μm , and 0.5 μm CMOS designs⁵. For each technology, we scaled the width of the driver transistor to 4, 6, 8, and 10 times its minimum width (which is a commonly used technique to speed-up the critical nets), and obtained four different values of R_d (and thus four different resistance ratios) for each of the technology. We generated 100 8-sink signal nets (uniformly distributed in a routing area of 0.5 mm x 0.5 mm) and routed them by the A-tree algorithm and the 1-Steiner algorithm, respectively. Figure 3 shows the improvement of the wiresized A-trees over the Steiner trees in term of the average delay for different technology and different transistor sizes.

The results show clearly that given a fixed technology, the improvement of the A-tree approach over the classical Steiner approach becomes more significant when the width of the driver transistor is increased, which results in lower resistance ratio. Moreover, we also observed a trend of increasing improvement by the A-tree algorithm as the device dimension decreases. For the conventional 2.0 μm CMOS technology, the A-tree algorithm in fact performs worse than the 1-Steiner algorithm. However, for the advanced 0.5 μm CMOS technology, the A-tree algorithm outperforms the 1-Steiner algorithm by a significant margin⁶. Since technological advances result in smaller and smaller device dimensions, we expect that our A-tree algorithm will achieve an even more significant improvement over the traditional Steiner approach.

6 Ongoing Research

We are further investigating the interconnect design problem in several directions. First, we shall extend the interconnect topology design problem to the case where the required time at each sink is not uniform (i.e. the sinks on the critical paths require a smaller delay and other sinks may have a longer delay). In this case, we can modify the A-tree algorithm by introducing “forbidden region” for each critical sink so that the critical sinks are connected directly (or almost directly) to

⁵The 2.0 μm , 1.5 μm , and 1.2 μm CMOS technology parameters are provided by Robit Foresight Inc., and the 0.5 μm CMOS technology parameter by MCNC.

⁶In general, the performance improvement obtained by the A-tree algorithm for IC technology is less than for MCM technology.

the source. Also, we are interested to develop a simple yet accurate delay model for distributed RLC circuits (where the non-monotonic circuit response presents a great difficulty) so that such model can be used effectively for interconnect design optimization when inductance is taken into consideration. Finally, we shall study the interconnect topology design and wiresizing problems under distributed RLC models so that we can consider the effect of reflection and cross-talk in the interconnect designs.

Acknowledgments

The authors would like to thank Dr. David Gao at Sun Microsystems and Prof. Wayne Dai at UC Santa Cruz for their helpful discussions and AT&T Microelectronics Division for providing the MCM parasitic parameters. This work is partially supported by National Science Foundation under grants MIP-9110511 and MIP-9110450.

References

- [1] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990, pp. 81-133.
- [2] 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.
- [3] 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.
- [4] J. Cong, K. S. Leung, and D. Zhou, “Performance-Driven Interconnect Design Based on Distributed RC Delay Model”, *UCLA Computer Science Tech. Report CSD-9200043*, Los Angeles, CA 90024, Sept. 1992.
- [5] W. Dai, Private communication, 1992.
- [6] 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.*, 1990, pp. 133-136.
- [7] W. C. Elmore, “The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier”, *J. Applied Physics*, 19(1948), pp. 55-63.
- [8] A. L. Fisher, and H. T. Kung, “Synchronizing Large Systolic Arrays”, *Proc. SPIE 341*, May 1982, pp. 44-52.
- [9] A. B. Kahng, and G. Robins, “A New Class of Iterative Steiner Tree Heuristics with Good Performance”, *IEEE Intl. Conf. on Computer-Aided Design*, July 1992, pp. 893-902.
- [10] 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*, 1987, pp. 518-519.
- [11] K. W. Lee, and C. Sechen, “A New Global Router for Row-Based Layout”, *IEEE Intl. Conf. on Computer-Aided Design*, 1988, pp. 180-183.
- [12] S. Prastjutrakul, and W. J. Kubitz, “A timing-driven global router for custom chip design”, *IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 48-51.
- [13] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor, “The Rectilinear Steiner Arborecence Problem”, *Algorithmica* 7 (1992), pp. 277-288.
- [14] J. Rubinstein, P. Penfield, and N. A. Horowitz, “Signal delay in RC tree networks”, *IEEE Trans. on CAD*, 2(3) (1983) pp. 202-211.
- [15] 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.
- [16] D. Zhou, S. Su, F. Tsui, D. S. Gao, and J. Cong, “A Distributive RCL-Model for MCM Layout”, *Proc. of IEEE Multichip Module Conf.*, March 1993.