An Iterative Approach for Delay-Bounded Minimum Steiner Tree Construction

Qing Zhu Mehrdad Parsa Wayne W.M. Dai

Board of Studies in Computer Engineering
University of California, Santa Cruz, CA 95064
qingz, courant, dai@cse.ucsc.edu

UCSC-CRL-94-39, Oct. 1994

Abstract

This paper presents a delay-bounded minimum Steiner tree algorithm. The delay bounds, given as inputs to the algorithm, can be different for each individual source-sink connection. The approach is based on feasible search optimization that satisfies the delay bounds first, then improves the routing tree for the cost minimization. Iterative cut-and-link tree transformation constrained by delay bounds provides an efficient technique to reduce the cost. Once reasonable delay bounds are set, this algorithm constructs Steiner trees with the correct timing, and by experiments the costs are always less than the trees obtained by a well-known, provably near-optimal Steiner-tree heuristic within the factor $2(1 - \frac{1}{|S|})$ of the *optimal* Steiner tree for $|S|$ sinks. In order to satisfy given delay bounds, we also propose a new algorithm to construct a maximal-delay-slack tree based on the global information of sink delay slacks. The use of our algorithm is especially attractive for deep-submicron VLSI layout or MCM substrate layout where the interconnect resistance is comparable to the driver on-resistance, when the minimal delay tree may have an unacceptably high cost while the traditional minimum Steiner tree fails the delay requirements.

Steiner tree, minimum Steiner tree, delay-bounded minimum Steiner tree, routing, delay-bounded routing, layout, timing-driven layout, shortest path, feasible search optimization

1 Introduction

As feature sizes shrink to 0.5 micron and less, we enter the era of deep-submicron VLSI designs. According to the interconnect scaling theory[3], the interconnect resistance is rapidly increased proportional to the square of the scaling factor, such that the interconnect resistance becomes comparable to gate on-resistance. Meanwhile, as the gate delays and gate sizes are scaled down, the interconnect delays become more dominant. As the circuit switching speed approaches to 200MHZ, the *wiring configurations* of critical nets and clock nets have to be carefully designed for correct timing. For the timing issues of deep-submicron designs, interconnect optimization is becoming just as important as device optimization.

Most existing placement and routing tools handle the design net-by-net, but timing requirements are expressed in the form of logic paths. Usually delay upper bounds are provided for logic paths based on the required clock frequency and critical path throughput speeds. In order to make use of existing place&route tools, delay bound of each logic path is divided into multiple bounds for pairs of source-to-sink logic cells on this path. Hauge, Nair and Yoffa [22] proposed the zero-slack algorithm that computes the maximal allowable delays for individual source-to-sink connections, based on entire logic path delay requirement. These maximal allowable delays for individual source-to-sink connections are then conveyed to place&route procedures. Works on delay bound generation can also be found in [31, 42, 18]. This delay-bounded net-by-net layout strategy has been used in the placement step in Cadence's Gate-Ensemble system [10] and Mentor Graphics's AutoCell system [42]. This paper provides the related work which can be used in the delay-bounded net-by-net routing.

1.1 Survey of Previous Work

Timing-driven placement algorithms have been proposed in [25, 38, 37, 15, 16, 9, 19]. Timing-driven routing algorithms were developed in recent years, since the traditional minimum Steiner routing tree may violate the timing requirements, especially for high performance layouts in deep submicrons or MCM substrates. Even if the placement has been timing-driven, the exact path delays are determined only after the routing trees are constructed.

People construct routing trees that trade-off the tree cost and the radius (the longest source-to-sink path length). One early related work is done by Ho, Lee, Chang and Wong[23] which investigated the complexity and NP-hardness of finding a bounded-diameter spanning tree or a bounded-diameter Steiner tree in a graph. Cohoon and Randall [11] proposed a heuristic which simultaneously considered the cost and the radius. Cong, Kahng, Robins, Sarrafzadeh and Wong [12] proposed a generalized formulation using a parameter ϵ to trade-off the radius and cost. They [12] proposed a provably good algorithm that can always generate a tree in which the path length is bounded by $(1+\epsilon)R$ (R is the source-to-sink radius) and the wiring cost is within a factor $2(1+(2/\epsilon))$ of the optimal Steiner tree. A similar approach is also proposed by Awerbuch, Baratz and Peleg [2]. Lim, Cheng and Wu [30] proposed a modified version of Prim's minimum spanning tree algorithm to control the radius for individual source-sinks connections. The radius-cost trade-off tree can be viewed as the one constructed between the minimum spanning tree (MST) (or minimum Steiner tree) and the shortest-path tree (SPT); Alpert, Hu, Huang and Kahng proposed a algorithm [1] which makes this MST-SPT combination in the tree construction. Cong, Leung and Zhou [13] optimized special Steiner arborescences called A-trees to make the MST-SPT trade off.

An important problem needed to be addressed is how to turn the delay bounds to the path length bounds or radius, since the delay of a RC tree depends on the topology of the tree. In deep submicron ICs, interconnect behaves distributed RC delays, and the path length is not accurate for the delay estimation [40].

Recently, minimal RC delay routing trees have also been investigated, which has three interesting variants: (1) minimizing the delay from the source to an identified critical sink or a set of critical sinks; and (2) minimizing the maximal source-to-sink delay; (3) minimizing the maximal delay slack. Boese, Kahng, McCoy and Robins have proposed several methods [5, 6, 7] to minimize the delays at identified critical sinks; i.e. the optimal tree minimizes the linear combination of sink delays $f = \sum_{i=1}^{k} \alpha_i \cdot d(s_i)$, where α_i ($\alpha_i \geq 0$) is the the criticality of the sink s_i. They revealed the good fidelity of using Elmore delay model[17] to guide a minimal delay routing tree compared to being guided by SPICE simulation. They proposed two branch and bound algorithms BB-SORT-C and BB-SORT. BB-SORT-C constructs the optimal critical sink tree to minimize the linear combination of sink delays, and the optimality is proved based on the observation [6] that the Hanan grid contains all Steiner points of the optimal critical sink tree. BB-SORT constructs the near-optimal tree that minimizes the maximal Elmore delay of the routing tree. Although BB-SORT-C and BB-SORT have exponential time complexity, they provides the optimal or near-optimal solutions for the empirical analysis of other heuristics. Boese, Kahng and Robins also proposed two heuristics [5, 7, 6] for the minimum Elmore delay Steiner trees: SERT and SERT-C. SERT is a modified Prim's algorithm when adding sinks to the tree it minimizes the maximal Elmore delay of the routing tree. SERT-C is a modification of SERT that minimizes the delay at a single critical sink. While Hong, Xue, Kuh, Cheng and Huang [24] proposed a modified Dreyfus-Wagner Steiner tree algorithm for minimizing the maximal path delay, Prasitjutrakul and Kubitz [33] proposed a routing algorithm for the minimization of the *delay slacks* (i.e., differences between the real delays and the given delay bounds) at sinks. Prasitjutrakul and Kubitz's algorithm will be described in more details in Section 4.

The major difficulty in the critical sink delay minimization is how to choose the accurate criticality for each critical sink to satisfy the delay bounds of multiple critical paths. Designers have to depend on intuition and perform design iterations. In typical deep submicron designs, more than 60 percent of the paths in a timing-critical design are critical [39].

1.2 Contribution of Our Work

Instead of using criticalities or radius for sinks, we use precise delay upper bounds which can be obtained from delay-bound-generators[22, 32] for meeting the logic path delay requirement. Most of previous timing-driven routing algorithms focus on the minimization of source-sink delays, except for [12] which provides the cost bounded within a factor $2(1 + (2/\epsilon))$ of the optimal solution. Delay issues should be considered in conjunction with the cost minimization. In deep submicron technologies, more and more functional blocks will be integrated into a chip. Interconnections, between functional blocks within a chip or chip-to-chip in a MCM, are not only timing critical, but also span a large area of layout that incurs high cost. This paper works on the delay bounded minimum Steiner tree that minimizes the cost of the routing tree under constraints of variable delay bounds for individual source-sink connections. The cost function can include not only the total wire length, but also the congestion and routing area. This work can also be applied on the delay balanced clock routing with the minimum wire length. Instead of zero skew routing, the clock net is routed as a delay bounded tree with a tolerable skew [44]. Taking advantage of the less stringent requirement of tolerable skew, we can significantly decrease the total wire length of the clock tree.

This paper presents a new algorithm that constructs a routing tree satisfying the delay bounds, while minimizing the cost of the tree. Although the model of delay-bounded minimum Steiner tree has been discussed in previous publications, we generalize the formulation of the problem to more realistic timing-driven layout. The major novelties of our algorithm, which we call Delay Bounded Minimum Steiner Tree (**DBMST**) approach, can be summarized as follows:

1. It accepts the delay bounds directly (not the path length bound or radius or sink criticality) in the tree construction. Furthermore, for sinks on different logic paths, the algorithm takes *variable* delay bounds.
2. Instead of using a single pass as in most previous algorithms, it iteratively minimizes the cost of the routing tree. The delay bounds are guaranteed based on feasible search optimization.
3. In order to find a feasible solution satisfying delay bounds as much as possible, the initial tree is constructed as a delay bounded tree that maximizes the delay slacks at sinks. We propose a new algorithm that constructs a maximum-delay-slack tree driven by the global information of delay slacks.

The remaining of this paper is organized as follows. Section 2 defines the delay bounded minimum Steiner tree problem. Section 3 outlines the entire optimization process of DBMST approach. Section 4 states a new max-delay-slack tree algorithm. Section 5 describes the cost reduction technique under delay bound constraints. Section 6 analyzes the complexity. Section 7 addresses DBMST's performance by simulation. Finally, Section 8 provides our conclusions.

specified).

3 Overview of DBMST Approach

The approach and optimization technique adopted here are based on an order that satisfies the delay bounds first, then improves the routing tree for the cost minimization. Hence, tree cost minimization is proceeded upon that the tree performance (delay requirement) has already been satisfied. Once reasonable delay bounds are set, DBMST outputs routing result in the correct timing and near-minimum cost.

DBMST uses the *feasible search optimization* technique to construct a Steienr tree, that minimizes the cost within a feasible region of delay bounded trees . Feasible search optimization is a very efficient method for constrained optimization problems, which have been widely applied in engineering fields [3]. Here, a feasible region R_f consists of routing trees that satisfy delay bounds given by the DBF. The basic idea is illustrated in Fig. 3, starting from an initial tree T_0 that is a delay-bounded Steiner tree, the technique examines a sequence of trees $T_1, T_2, \ldots$, and terminates at a delay-bounded Steiner tree T_n with the least cost.

DBMST consists of two major steps; a skeleton flowchart is shown in Fig. 3(b). Step 1 constructs T_0, then Step 2 transforms the current tree T_i to T_j, where $cost(T_i) > cost(T_j)$. The delay slack at each sink is the deduction of the real delay from the delay bound. In order to let T_0 be in the feasible region as a delay bounded tree, we maximize the delay slack when constructing T_0; T_0 is a maximal-delay-slack tree.

In some cases, the delay bounds given by DBF may be too tight, i.e., they cannot be met even in T_0. Ensuring the correct timing, delay violations at the routing stage are back-annotated to the stages of placement or logic synthesis, measures being taken as to update the locations of sinks or the source or enlarge the delay bounds of specific sinks. This back-annotation procedure is indicated in Fig. 3(b). The rest of this paper assumes that DBF gives delay bounds that can be met by T_0.

---

[3] Brayton, Hachtel and Sangiovanni-Vincentelli [8] gives a good survey of feasible search optimization methods.

2. **Critical mapping**: we select a mapping with the least weight.

The *mapping* step determines the best path for each free sink once added to the tree. Recall that we have assigned the weight of an edge in G as the minimum delay slack of the resulted tree, when the free sink (in V_r) is connected to a specific tree node (in V_l). So, the mapping step chooses the connection pattern of a free sink in the objective that maximizes the minimum delay slack of the resulted tree (including all connected sinks). In a short, this step determines the best branching point on the tree to connect the free sink. As shown in Fig. 6(a), we obtain the mapping result of free sinks for the example shown in Fig. 5. A shortest path is preferred from the branching point connected to the free sink, since the shortest path adds the minimum additional capacitive load to already connected sinks such that the increase of delays at these connected sinks are minimized (see Fig. 5(a)). In addition, comparing to longer paths from the same branching point on the tree, a shortest path always minimizes the delay to the chosen free sink. If multiple shortest paths exist, the path that passes through the minimum number of other sinks is always selected.

The *critical mapping* step determines which free sink is connected to the tree next. The critical mapping at one stage of tree growing is the mapping with the least weight. The free sink of the critical mapping is called *critical free sink*. In other word, the critical free sink is selected of all free sinks that will results in the smallest delay slack of the tree if this critical free sink is connected. As illustrated in Fig. 6(b), t_4 to n_1 mapping is the critical mapping, since it has the least mapping weight, i.e. $w(t_4, n_1) = min(w(t_3, n_3), w(t_4, n_1), w(t_5, n_1))$, and t_4 is the critical free sink. The resulted tree by the critical mapping is shown in Fig. 6(d).

The algorithm always picks up the *critical* free sink next connected to the tree, using the path decided by the critical mapping. The reasons are as follows. As we know, for a sequential router (everytime a destination is connected), early stage connections have more flexibility to maximize the delay slacks of sinks, and later connections have less freedom or optimization space due to the existence of connected sinks. Furthermore, the optimality of the routing tree depends heavily on the maximization of delay slacks of critical sinks.

The algorithm starts from the source, and at each stage a critical free sink is connected to the routing tree using the critical mapping. An example is illustrated in

Fig. 14(b), where T_0 is constructed using the algorithm presented above; and terminals are marked in the sequence of connections to the tree (i.e. t_1 is connected before t_2, etc.).

The high-level pseudo code of the max-delay-slack tree algorithm are described in Fig. 7.

```
INPUT :
    $G(V, E)$ = graph, $s$ = source,
    $S$ = set of sinks, $n$ = number of sinks,
    $DB$ = set of delay bounds for sinks.
OUTPUT :
    A max-delay-slack Steiner tree spanning $S \cup \{s\}$.
PROCEDURE  MDST ($G(V, E)$, $s$, $S$, $DB$, $n$) {
    $i = 0$;
    $T = \{s\}$, $V_l = \{s\}$, $V_r = S$;
    while  ($i < n$) {
        Update the weighted bipartite graph $G = (V_l, V_r, E)$
                $V_l$ = set of nodes on the tree, $E(V_l, V_r)$ = set of shortest paths between $V_l$ and $V_r$,
                $W(E)$ = set of weights of $E$ (related to delay slacks);
        Map each free sink in $V_r$ to the tree node in $V_l$ with the largest weight;
        $t^*$ = critical sink in $V_r$ with the least mapping weight;
        Connect $t^*$ by the critical mapping;
        $T = T + t^*$, $V_r = V_r - t^*$, $i = i + 1$;
    }
}
```

Figure 7: Max-Delay-Slack Tree Algorithm Description

In order to reduce the computational cost, we make a *incremental update* of the bipartite path slack graph G once a new free sink is connected. The incremental update of G is accomplished in following three steps:

(a) Add on G new nodes of the tree T;

(b) Update connection paths from nodes of the tree to free sinks;

(c) Update weights of edges in G.

The key step of the incremental update is step (b). Step (a) is done by listing new nodes on the left side of G; these new nodes are just added to T. On the right side

The nice property of using the above incremental shortest path construction, is that once a shortest path is found satisfying DBF, the path is definitely the delay bounded *shortest* path between T^a and T^b. Define k as the number of shortest paths constructed in the incremental constructions before we find the delay bounded one. We found in experiments that usually the k is very small once we obtain a delay bounded shortest path. Three reasons make the k small to get a delay bounded shortest path: (a) we stop the incremental shortest path construction, immediately when we detect the current shortest path resulting in a delay bounded tree; (b) we stop the construction once the current shortest path has larger cost than the deleted path from T; (c) in most cases, a shorter path between T^a and T^b helps the delay bound satisfaction of the entire tree. These above reasons make the delay bounded shortest path is found at early stage of the incremental shortest path construction. The k-shortest paths can be constructed in $O(kn^2)$ [28] for n grids in the routing graph. On the other hand, if we statically obtain all pairs of shortest paths between T^a and T^b, the time complexity will be $O(|T|n^2)$ where $|T|$ is the size of the routing tree.

We show the high-level description of the DBMST algorithm in Fig. 13. To maximize the cost reduction, the stop criterion of DBMST is devised such that the algorithm terminates when none of cut-and-link paths results in the positive cost reduction. The convergency is guaranteed by the monotonical cost reduction of tree transformations.

Fig. 14 illustrates steps of constructing a routing tree with the requirement of variable delay bounds. We obtain the result using 0.3um CMOS technology given in Section 7. The tree in Fig. 14(h) is constructed by DBMST, and the tree in Fig. 14(e) by a minimum Steiner tree heuristic proposed by Kou, Markowsky and Berman (KMB) [27]. We show the first five stages of tree transformation $(T_0, T_1, T_2, T_3, T_4, T_5)$ in Fig. 14. As shown in Fig. 15(a), the tree constructed by DBMST satisfy delay bounds at sinks, while KMB has no knowledge of path delay bounds so as to fail at 5 sinks. DBMST reduces the cost by 8% compared to KMB, although KMB algorithm has achieved near-optimum cost Steiner tree [27] [4]. The significance of DBMST is that it accomplishes the routing not only satisfying the required delay bounds but also close to the minimum cost.

---

[4] KMB algorithm is proved [27] to construct a Steiner tree with a tight cost $C_{KMB} \leq 2(1 - \frac{1}{|S|})C_{opt}$, where C_{opt} is the cost of the *optimum* Steiner tree, and $|S|$ the number of sinks.

```
INPUT:
    $G(V, E)$ = graph, $s$ = source,
    $S$ = set of sinks,
    $DB$ = set of delay bounds for sinks,
OUTPUT:
    A delay bounded Steiner tree spanning $S \cup \{s\}$.
PROCEDURE DBMST($G(V, E)$, $s$, $S$, $DB$) {
    $k = 0$;
    $T_0$ = max-delay-slack tree spanning $S \cup \{s\}$ that maximizes delay slacks of sinks;
        (using algorithm described in the next section);
    do {
        $BestGain = -\infty$;
        Derive the deducted tree $T'_k$ from $T_k$;
        for (Each path $p$ in $T_k$ (each edge in $T'_k$)) {
            $q$ = delay bounded shortest path corresponding to $p$;
            $g$ = gain if $p$ is switched to $q$;
            if ($BestGain < g$)
                $BestGain = g$;
        }
        if ($BestGain > 0$) {
            $(p, q)$ = pair of cut-and-link paths with the gain equal to $BestGain$;
            Remove $p$ from tree $T_k$, making $T_k - p$ equal to two subtrees $T^a$ and $T^b$;
            $T_{k+1} = q + T^a + T^b$;
            $k = k + 1$;
        }
    } while ($BestGain > 0$);
}
```

Figure 13: DBMST Description

never larger than the cost c of the deleted path. So, gain $g = c - c^* \geq 0$. $\square$

Since we select the maximal gain of cut-and-link paths at each tree transformation, based on Lemma 2, we have theorem 2.

Theorem 2: *DBMST decreases the tree cost monotonically in the steepest gradient search direction.*

5.3 Cost Optimality

Constructing a delay bounded minimum Steiner tree is a NP-complete problem in the aspect of the cost minimization. DBMST is a good heuristic for delay bounded minimum Steiner tree problem, not only it satisfies the variable delay bounds during the entire construction process (Appendix 2 shows that constructing a tree satisfying delay bound constraints itself is a NP-complete problem), but also turns the tree out with reasonable small cost.

In all experiments we have done so far, DBMST always achieves the cost less than the known minimum Steiner tree heuristic proposed by Kou, Markowsky and Berman (KMB), which has no delay bounds. KMB algorithm has a tight bound of the cost to the optimum Steiner tree [27], i.e. $C_{KMB} \leq 2(1 - \frac{1}{|S|})C_{opt}$ [5], where C_{KMB} is the cost of the Steiner tree constructed by KMB algorithm, C_{opt} the cost of the optimum Steiner tree, and $|S|$ the number of sinks.

We can improve DBMST in the cost minimization by the following techniques (a) Search the tree configurations out of the feasible search region for increasing the opportunity of obtaining the global optimum; (b) Hill-climb the local minimum by back-tracking to some earlier stage tree configurations, and try transforming the tree in a different search direction instead of the maximum gain steepest-gradient direction (for example, selecting the tree oriented from the cut-and-link with the second maximal gain). We explain the first technique using the *group migration* principle as follows.

Always limiting the search in the feasible region (R_b) definitely looses some opportunity of tracing the best search path to the global optimum solution. The opportunity

---

[5] Empirical analysis shows that DBMST achieves less cost than KMB algorithm which further obtains smaller cost bound than the bounded radius routing tree algorithm in [12] that achieves a factor of $2(1+(2/\epsilon))$ of the optimum Steiner tree.

we first try to find a cut-and-link in R_b with the maximal gain. If no cut-and-link is allowable in R_b due to the delay bound requirement, we continue to find a cut-and-link in R'_b with the maximal gain using the same algorithm as in Section 4.2. The cut-and-link in R'_b is accomplished such that the routing tree is allowed to have a relaxed negative minimum slack, i.e. $s^* \geq -\delta$ ($\delta \leq 0$); when we decide the delay bounded shortest path p between two subtrees T^a and T^b, p is the shortest one that makes the routing tree $T = T^a + T^b + p$ with the smallest slack$s^* \geq -\delta$.

A transition cost $c_{k,k+1}$ from T_k to T_{k+1} is defined as follows, where C_{k+1} and C_k are costs of T_{k+1} and T_k.

$$c_{k,k+1} = \begin{cases} C_{k+1} - C_k & \text{if } T_{k+1} \text{ is a delay bounded tree} \in R_b; \\ +1 & \text{else } T_{k+1} \in R'_b. \end{cases} \tag{4}$$

Based on Lemma 2, if T_{k+1} is a delay bounded tree, always $c_{k,k+1} \leq 0$. Based on (4), if T_{k+1} not satisfying the delay bounds but still with $s^* \geq -\delta$ (in R'_b), always $c_{k,k+1} = +1$ (positive). As shown in Fig. 16(b), the first step of cut-and-link has positive $c_{k,k+1}$ (the delay bounds are not satisfied), but we still proceed to the following $m-1$ cut-and-link trees, and select m^* cut-and-link tree transformations as the result of T_{k+1}; T_{k+1} has less cost than T_k since $c^*_{k,k+1} < 0$. If a sequence of m cut-and-link trees results in positive (+1) transition cost at every time, as shown in Fig. 16(c), then $T_{k+1} = T_k$, because no new delay bounded T_{k+1} can be constructed by these cut-and-link trees.

Relaxing the exact delay bounds in the search of intermediate tree configurations, or allowing local search in a expanded pseudo feasible region R'_b, probably provides more opportunity to execute further cut-and-link for more cost reduction. The final tree selected in m-cut-and-link is always a delay bounded tree that is retained in feasible search region R_b, that is a starting point for continuing cut-and-link tree transformations.

6 Complexity

We analyze the time complexities of two major stages of DBMST: constructing T_0 and later optimization process.

At each stage of constructing the max-slack-tree (initial T_0), in order to build the bipartite graph, we iterate each node on the routing tree T and find the shortest path tree in $O(n^2)$, where $n = |V|$. Since m sinks are connected, the total computational time of T_0 is $O(m|T_0|n^2)$, where $|T_0|$ is the number of nodes in the output tree. The dynamic update of the bipartite path slack graph can further reduce the computational time. Once updating the bipartite graph, the times of re-computing shortest paths is proportional to the size $|P^*|$ (number of nodes) of the new path p^* added to the tree, instead of proportional to entire $|T|$. During the entire period connecting m sinks, the sum of sizes $|P^*|$ of new paths is exactly equal to the size of the output T_0 (every time a new path is added to T_0). So, the overall time complexity of T_0 construction, by the dynamic updating of the bipartite path slack graph, is $O(|T_0|n^2)$.

The computation of the tree transformation in feasible search optimization is dominated by computing the delay bounded shortest path which can be done in $O(kn^2)$ [28] when using incremental construction of pairs of shortest paths, where k is the number of shortest paths constructed for finding the delay bounded shortest path. Usually k is a very small number (close to 1 or 2), once a delay bounded shortest path is found (see the analysis in Section 4). In order to obtain the path switching using the maximal gain of the cost reduction, we iterate paths in tree T. So, a tree transformation in the optimization is accomplished in $O(k|T|n^2)$, where $|T|$ is the edge number of the tree.

7 Experimental results

DBMST has been implemented in C++. In our experiments, the cost of solutions of DBMST is evaluated and is compared to the cost of solutions obtained by the KMB algorithm [27], the KMB algorithm followed by iterations of cut-and-link improvement, and the minimum-delay SERT algorithm of Boese and Kahng et al [7]. The experiments are performed on random circuits. The random circuits were constructed by uniformly sampling locations of sinks from a rectangular region. The Manhattan distance between the points is used as the cost function, and the Elmore delay as the delay model for the circuits.

The technology parameters for the delay model are given in Table 1. The values for the 1μm CMOS are obtained from MOSIS. The values for the 0.5 and 0.3 μm

CMOSs are extrapulated from the 1μm CMOS using scaling formulas [3]. That is, as the feature size scales by a factor S, the resistance per unit length scales as S^2, the capacitance per unit length remains constant, and the gate capacitance is decreased by $1/S$. The thin-film MCM values are from AT&T Microeletronics Division.

	chip size	R_b $(m\Omega/\mu m)$	C_b $(fF/\mu m)$	R_d (Ω)	C_t (pF)
1 μm CMOS	10 × 10 mm^2	30	0.02	100	0.02
0.5 μm CMOS	20 × 20 mm^2	120	0.02	100	0.01
0.3 μm CMOS	40 × 40 mm^2	480	0.02	100	0.005
MCM	100 × 100 mm^2	8	0.06	25	0.2

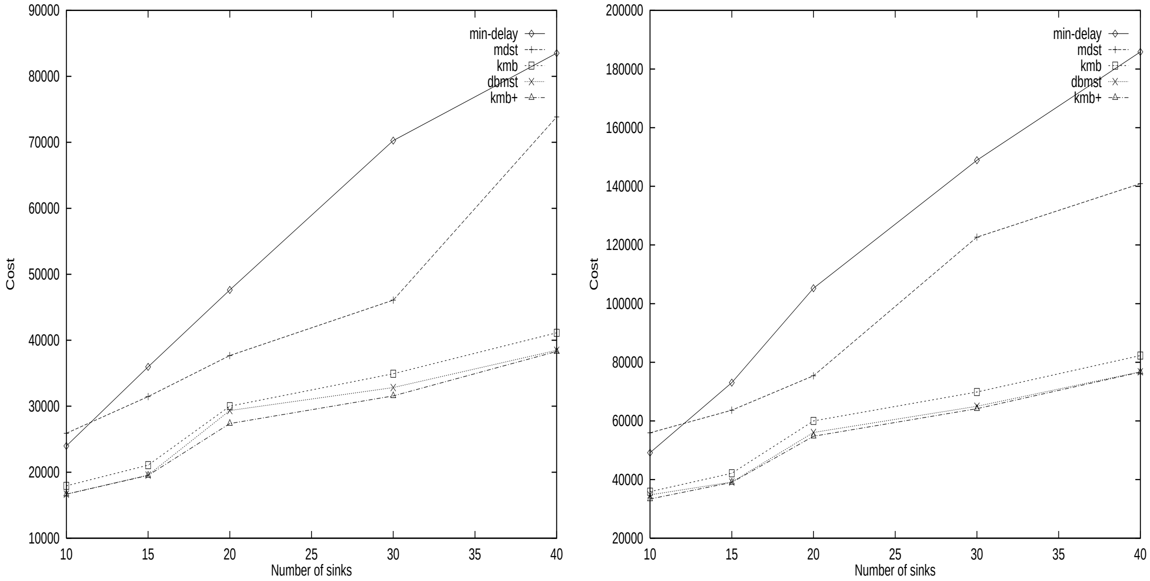
Table 1: The technology parameters used in the experiments.

The experimental results for different size nets and technologies are shown in Fig. 17 and Fig. 18. The curve labeled `min-delay` corresponds to the cost of the minimum-delay tree algorithm of Kahng et al. [6, 7]. The `kmb` curve is produced by the KMB algorithm, and `kmb+` curve is produced by applying the path-switching technique in `dbmst` to the result of the KMB algorithm. Finally, the `dbmst` curve corresponds to the solution of DBMST. These notations are listed in Table 2 for reference.

MDST	proposed heuristic of max-delay-slack tree, as the initial DBMST
`mdst`	curve or result of MDST
DBMST	proposed heuristic of delay bounded minimum Steiner tree,
	that improves the tree by iterative delay bounded cut-and-link
`dbmst`	curve or result of DBMST
KMB	heuristic of minimum Steiner tree [27]
`kmb`	curve or result of KMB
KMB+	heuristic of improving KMB, followed by iterative cut-and-link improvement
`kmb+`	curve or result of KMB+
MIN-DELAY	heuristic of minimum Elmore delay Steiner tree (SERT)[7]
`min-delay`	curve or result of MIN-DELAY

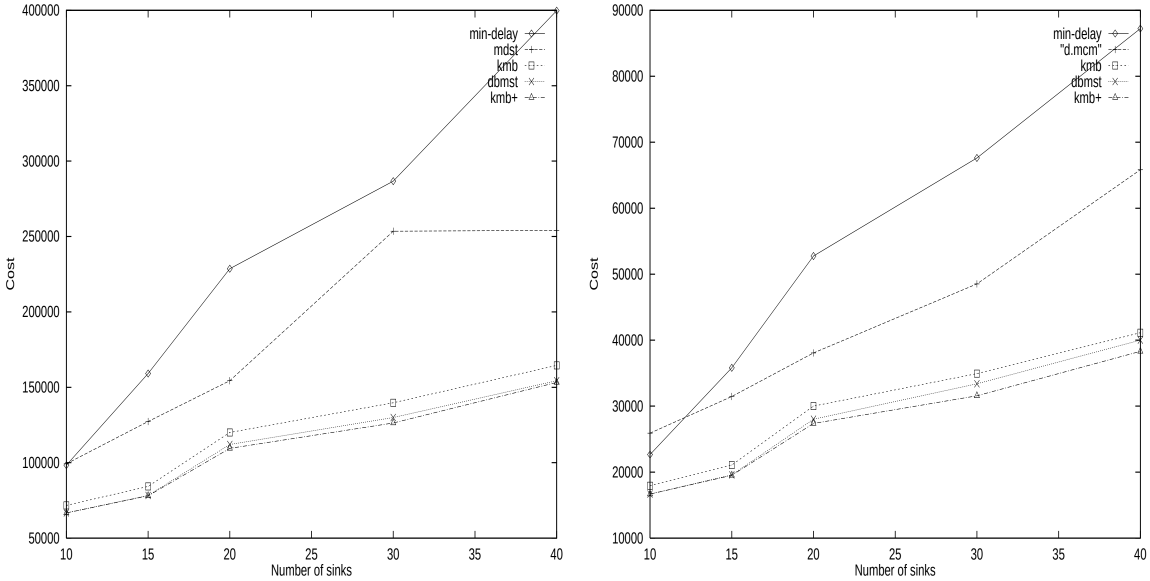
Table 2: List of Notations

For the sake of comparing the cost with the minimum delay tree, the delay bound is set to be the same for all the sinks, and it is the maximum delay of the solution of the KMB algorithm. The purpose here is show that if the delay bound is as loose as that of the KMB algorithm, the cost of DBMST is as small as that of the KMB algorithm, while the delay bound is satisfied at all times. A tighter delay bound trades off the minimum cost achievable by DBMST.



(a) (b)

Figure 17: Comparison of costs. (a) 1μm CMOS. (b) 0.5 μm CMOS.



(a) (b)

Figure 18: Comparison of costs. (a) 0.3μm CMOS. (b) thin-film MCM.

Iterative cut-and-link technique is shown to be very efficient to reduce the cost.

Table 3 shows the cost decrease percentage due to the iterative cut-and-link. The cost percent decrease from `kmb` to `kmb+` is about 8.5%, that suggests the iterative cut-and-link technique can be significantly applied on any Steiner tree constructed by some heuristic for the further improvement. The significance of iterative cut-and-link technique on the cost reduction of DBMST is even more, shown in Table 3, that the cost reduction from intial MDST to DBMST is about 60.5%. Although the delay bounds are set as constraints of path switching, DBMST achieves less cost than KMB for all testing examples, and is very close to the cost of KMB+. On average in Table 4, DBMST has only 1.5% more cost than $KMB+$ on testing examples. Meanwhile, the cost of the minimum-delay tree is almost twice as large; thus not taking advantage of the available delay slack will lead to unnecessarily costly results. Table 4 shows the relative costs of the DBMST and minimum-delay trees with respect to the KMB+ trees for the different technologies.

	`kmb` → `kmb+`	`mdst` → `dbmst`
1 μm CMOS	8.7%	57%
0.5 μm CMOS	8.3%	69%
0.3 μm CMOS	8.7%	64%
MCM	8.7%	52%

Table 3: Cost Percent Decrease by Iterative Cut-and-Link.

Results in Table 4 shows that as the feature size becomes smaller accompanied with the enlarged chip size, the ratio of the cost of the minimum-delay tree to the minimum-cost tree increases; therefore, the cost issue become more dominant as the feature size decreases with the current technology trend. The cost and performance is also more significant for nets with large spans and heavy loads. This is of practical importance in routing clock trees driven by the tolerable skew so as to minimize the cost penalty. For DBMST as shown in Table 4, cost ratio remains almostly constant to the minimum-cost tree.

As an indication of the computational demand of DBMST, its execution time on a SUN Sparc1+ is shown in Fig. 19. It was observed during the experiments that for larger size nets, the construction of the initial feasible maximum-delay-slack tree dominates the execution time. Nevertheless, the execution time is rather fast and makes

	`kmb+`	`dbmst`	`min-delay`
1 μm CMOS	1	1.02	1.94
0.5 μm CMOS	1	1.02	2.09
0.3 μm CMOS	1	1.01	2.18
MCM	1	1.01	1.98

Table 4: The Relative Costs Normalized with Respect to `kmb+`.

DBMST very practical. Test examples covering up to 40 sinks provide the availibility of DBMST used in the clock routing by tolerable skew bounded.

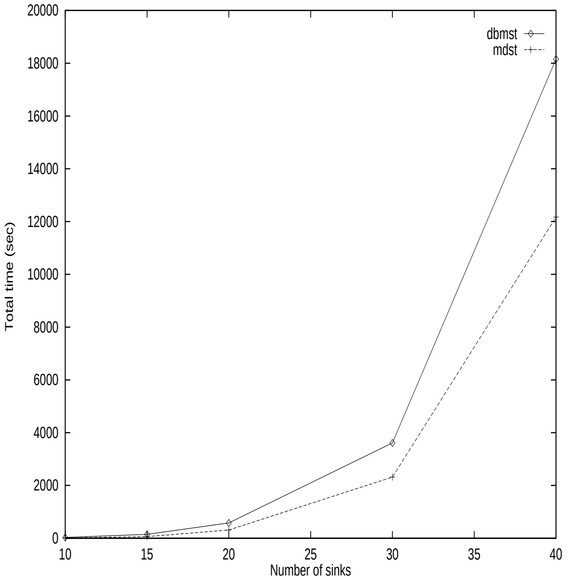


Figure 19: CPU Time of `dbmst` and Initial `mdst` on Test Circuits. The data is measured on a SUN Sparc1+.

8 Conclusions and Further work

We have presented a delay bounded minimum Steiner tree construction algorithm. It accepts variable delay bounds for individual source-sink connections, and minimizes the cost of the tree within the constraint of the delay bounds. The methodology and optimization technique is adopted based on a natural order that satisfies the delay bounds first, then improve the routing tree for the cost minimization. Once reasonable delay bounds are set, this algorithm outputs routing result in the correct timing and near-minimum cost. This algorithm is especially attractive to the layout design when the interconnect resistance approaches to the gate on-resistance in deep-submicron VLSIs or on the substrate of thin-film multi-chip modules.

We will investigate the further application of this algorithm on the routing of timing critical nets, such as the clock net routing. Once we set a tolerable skew, we intend to route the low levels of the clock tree with delay-bounded minimum Steiner tree for the reduction of the total wire length.

The correctness of the proposed routing algorithm is independent of the delay model used, although, the *optimality* in the electrical performance and cost of the resulted tree is related to the delay model. High order (at least second-order) delay models [34, 29] are necessary for the high-frequency interconnect, especially on the substrates of multichip modules or boards once the transmission line effects are noticed.

9 Acknowledgement

This work was supported partially by National Science Foundation Presidential Young Investigator Award under Grant MIP-9009945. We would like to thank Kenneth Boese and Prof. Andrew Kahng at UC Los Angeles for providing their source code [7].

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. *Proceedings of IEEE Intl. Symposium on Circuits and Systems*, 1993.

[2] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensetive analysis of communication protocols. In *Proc. ACM Symp. on Principles of Distributed Computing*, pages 177–187, 1990.

[3] H. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company, 1987.

[4] J. Benkoski and A. J. Strojwas. The role of timing verification in layout synthesis. In *Proc. of 29th Design Automation Conf.*, pages 612–619, 1992.

[5] K. D. Boese, A. B. Kahng, B.A. Mccoy, and G. Robins. Near-optimal critical sink routing tree constructions. In *technical report TR-930027, UCLA CS Department*, pages 1–43, 1993.

[6] K. D. Boese, A. B. Kahng, B.A. Mccoy, and G. Robins. Rectilinear steiner trees with minimum elmore delay. In *Proc. of 31th Design Automation Conf.*, pages 381–386, 1994.

[7] K. D. Boese, A. B. Kahng, and G. Robins. High-performance routing trees with identified critical sinks. In *Proc. of 30th Design Automation Conf.*, pages 182–187, 1993.

[8] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli. A survey of optimization techniques for integrated circuits. *Proc. of the IEEE*, 69(10):1334–1362, 1991.

[9] M. Burstein and M. Youssef. Timing influnced layout design. In *Proc. of 22nd Design Automation Conf.*, pages 124–130, 1985.

[10] A. H. Chao, E. M. Nequist, and T. D. Vuong. Direct solution of performance constraints during placement. In *Proc. of IEEE Custom Integrated Circuits Conference*, pages 17.2.1 – 27.2.4, 1990.

[11] J. P. Cohoon and L. J. Randall. Critical net routing. In *Proc. IEEE Intl. Conf. on Computer Design*, pages 174–177, 1991.

[12] J. Cong, A.B. Kahng, G. Robins, M. Sarrafzadeh, and C.K. Wong. Provably good performance-driven global routing. *IEEE Trans. on Computer-Aided Design*, CAD-11(6):739–752, 1992.

[13] Jason Cong, Kwok-Shing Leung, and Dian Zhou. Performance-driven interconnect design based on distributed rc delay model. In *Proc. of 30th Design Automation Conf.*, pages 606–611, 1993.

[14] W. W.-M. Dai, T. Asano, and E.S. Kuh. Routing region definition and ordering. *IEEE Trans. on Computer-Aided Design*, CAD-4(3):189–197, 1985.

[15] 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 *Proceedings of ACM/IEEEE Design Automation Conference*, pages 84–89, 1990.

[16] A.E. Dunlop, V.D. Agrawal, D. Deutsch, M.F. Jukl, P. Kozak, and M. Wiesel. Chip layout optimization using critical path weighting. In *Proceedings of ACM/IEEEE Design Automation Conference*, pages 133–136, 1984.

[17] W.C. Elmore. The transient response of damped linear network with particular regard to wideband amplifiers. *Jour. of Applied Physics*, 19:55–63, 1948.

[18] J. Frankle. Iterative and adaptive slack allocation for performance-driven layout and fpga routing. In *Proc. of 29th Design Automation Conf.*, pages 536–542, 1992.

[19] T. Gao, P.M. Vaidya, and C.L. Liu. A new performance driven placement algorithm. In *Digest of Tech. Papers of IEEE Intl. Conf. on Computer Aided Design*, pages 328–331, 1990.

[20] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NPCompleteness.* New York: W.H. Freeman and Co., 1979.

[21] M. Hanan. On steiner's problem with rectilinear distance. *SIAM Journal of Applied Mathematics*, pages 255–265, March 1966.

[22] P.S. Hauge, R. Nair, and E.J. Yoffa. Circuit placement for predictable performance. In *Digest of Tech. Papers of IEEE Intl. Conf. on Computer Aided Design*, pages 88–91, 1987.

[23] J. Ho, D. T. Lee, C. H. Chang, and C. K. Wong. Bounded-diameter spanning tree and related problems. In *Proc. ACM Symp. on Computational Geometry*, pages 276–282, 1989.

[24] X. Hong, T. Xue, E.S. Kuh, C.K. Cheng, and J. Huang. Performance-driven steiner tree algorithms for global routing. In *Proc. of 30th Design Automation Conf.*, 1993.

[25] M. A. B. Jackson and E. S. Kuh. Performance-driven for of cell based ic's. In *Proc. of 26th Design Automation Conf.*, pages 370–375, 1989.

[26] W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[27] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Information*, 15:141–145, 1981.

[28] E. Lawler. *Combinational Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[29] H. Liao, W. Dai, R. Wang, and F.Y. Chang. S-parameter based macro model of distributed-lumped networks using exponentially decayed polynomial function. In *Proceedings of 30th ACM/IEEE Design Automation Conference*, pages 726–731, 1993.

[30] Andrew Lim and Siu-Wing Cheng. Performance oriented rectilinear steiner trees. In *Proc. of 30th Design Automation Conf.*, pages 171–176, 1992.

[31] W. K. Luk. A fast physical constraint generator for timing driven layout. In *Proc. of 28th Design Automation Conf.*, pages 626–631, 1991.

[32] R. Nair, C.L. Berman, P.S. Hauge, and E.J. Yoffa. Generation of performance constraints of layout. *IEEE Trans. on Computer-Aided Design*, CAD-8:860–874, 1989.

[33] Somchai Prasitjutrakul and William J. Kubitz. A timing-driven global router for custom chip design. In *Digest of Tech. Papers of IEEE Intl. Conf. on Computer Aided Design*, pages 48–51, 1990.

[34] Curtis Ratzlaff, Nanda Gopal, and Lawrence Pillage. Rice: Rapid interconnect circuit evaluator. In *Proceedings of 28th ACM/IEEE Design Automation Conference*, pages 555–560, 1991.

[35] J. Rubinstein, P. Penfield, and M. A. Horowitz. Signal delay in rc tree networks. *IEEE Trans. on Computer-Aided Design*, CAD-2(3):202–211, 1983.

[36] T. Sakurai. Approximation of wiring delay in mos-fet lsi. *IEEE Journal of Solid-State Circuits*, SC-18:418–426, 1983.

[37] A. Srinivasan, K. Chaudhary, and E.S. Kuh. Ritual: A performance driven placement algorithm for small-cell ics. In *Digest of Tech. Papers of IEEE Intl. Conf. on Computer Aided Design*, pages 48–51, 1991.

[38] A. Srinivasan, M.A.B. Jackson, and E.S. Kuh. A fast algorithm for performance driven placement. In *Digest of Tech. Papers of IEEE Intl. Conf. on Computer Aided Design*, pages 328–331, 1990.

[39] M.Y. Mike Tsai and R.S. Tsay. Ic layout shift at deep-submicron level. *Electronic Engineering Times*, 820:66, Oct. 1994.

[40] R.S. Tsay. An analytic net weighting approach for performance optimization in circuit placement. In *Proc. of 28th Design Automation Conf.*, pages 620–625, 1991.

[41] A. Vittal and M. Marek-Sadowska. Minimal dealy interconnect design using alphabetic trees. In *Proc. of 31th Design Automation Conf.*, pages 392–396, 1994.

[42] H. Youssef, R.B. Lin, and E. Shragowitz. Bounds on net delays for vlsi circuits. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, 39(11):815–824, 1992.

[43] Qing Zhu and Wayne W.M. Dai. Delay-bounded steiner tree algorithm for performance-driven layout. *Technical Report, UCSC-CRL-93-46, University of California, Santa Cruz*, Oct 1993.

[44] Qing Zhu, Joe G. Xi, Wayne W.M. Dai, and Rama Shukla. Low power clock distribution based on area pad technology for multichip modules. In *Intl. Workshop on Low Power Design*, pages 87–92, 1994.

APPENDIX I

In VLSI chips, a RC tree is usually used to model the equivalent circuit Of a interconnect tree. Models of basic elements of the tree are shown in Fig. 20. We model an edge (wire) of the routing tree as a distributed RC line. For an edge e_i with length l_i, the line resistance is $r_i = r_s l_i$ and line capacitance $c_i = c_s l_i$, where r_s is unit length resistance and c_s the unit length capacitance. If wires on different metal layers, a via is modeled as a distributed RC line. Source driver has an on-line resistance R_d [7] and an output capacitance C_d, and each sink (terminal) has a load capacitance. Fig. 21 combines electrical elements together for the model of a routing tree, with no via in the same layer routing.

In the experiments, we use Elmore delay as the evaluation of the RC delay from the source to sinks. Elmore delay is the first-order approximation of the response of a step voltage input[17, 35]. Formally, the delay $d(s,t)$ from source s to a sink t is calculated as follows:

$$d(s,t) = R_d(C_d + C_0) + \sum_{e_i \in path(s,t)} r_i(c_i/2 + C_i) \tag{5}$$

where e_i is the edge from node n_i to its parent, C_0 the total capacitance of sinks and edges of the routing tree T, and C_i the total capacitance of sinks and edges in the subtree of T rooted at n_i.

The correctness of the routing algorithm proposed in this paper is independent of the delay model used (although, the *optimality* in the electrical performance and cost of the routing tree is related to the delay model). In a short, the algorithm can be accomplished using any form of delay model as the path delay function. High order (at least second-order) delay model [34, 29] is necessary for the high-frequency interconnect, especially on the substrates of multichip modules or boards once the transmission line effects are noticed.

---

[7] The on-line resistance for a CMOS inverter is the half-approximation of PMOS transistor and NMOS transistor.

APEENDIX II

Theorem: *Given a graph $G = (V, E)$, with the path delay function, a source s, sink set S, and a delay-bound function, finding the maximum delay slack tree problem is NP-complete.*

Proof: The problem is in NP, since a "guess" can list a set of edges that form the tree, and in deterministic time, it is possible to check:

1. the edges do form a tree
2. the nodes of S are all covered

The problem is NP-hard. Assume there exists a deterministic polynomial time algorithm A for the problem. Boese and Kahng [6, 7] formulated the *minimum delay tree problem* on an identified critical sink. Then given a minimum delay tree problem, we can construct a max-delay-slack tree problem as follows. Let the delay bound be D for the critical sink and ∞ for other sinks. A solution given by A is exactly the solution for the minimum delay tree on the identified critical sink. Thus, the optimal max-delay-slack tree problem is NP-complete, since the corresponding critical sink minimum delay tree problem is NP-complete [6].