# An Interconnect Topology Optimization by a Tree Transformation

Naofumi Tsujii, Katsutoshi Baba*, and Shuji Tsukiyama

Dept. of Electrical and Electronic Eng., Chuo University
1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, JAPAN

tsun@tsuki.elect.chuo-u.ac.jp tsuki@elect.chuo-u.ac.jp

∗ Currently, Fujitsu Corp., Japan

**Abstract — Since the interconnect delay has become the dominating factor in circuit performance, demands for a good delay–minimization router are very high. In this paper, we propose two algorithms to find an interconnect tree of a net which minimizes a weighted sum $\tau$ of delays to all sinks, where the weight assigned to a sink represents a criticality of the delay to the sink. The algorithms start from a Steiner tree and repeat a tree transformation as monitoring the change of $\tau$. We also show some experimental results to evaluate the performance of the algorithms.**

## I. Introduction

According to the rapid decreasing of the technology feature size, the interconnect has become a dominating factor in determining circuit performance. Hence, requirements to the interconnect design become severe and severe. To cope with this, many interconnect techniques have been proposed [1], such as wire length minimization, device sizing, interconnect topology optimization, buffer insertion, and wire sizing. Among them, interconnect topology optimization becomes more important, when the resistance ratio, defined as the driver resistance over the unit wire resistance, becomes small.

When the resistance ratio is large, Steiner tree[2, 3, 4], with the shortest total wire length has the minimum interconnect capacitance so that it gives an optimal interconnect topology. However, with the increase of the unit wire resistance, it cannot be optimal anymore. Therefore, several concepts have been proposed as an interconnect tree, such as Shortest Steiner Arborecence [5], cost-radius balanced tree[6], Steiner Elmore Routing Tree (SERT) or SERT-C (C stands for critical sink)[7], etc. Among them, SERT-C is a practical method, in the sense that it distinguishes critical sinks from other sinks in the interconnect tree design, where a critical sink is a sink of a net such that the path from the source of the net to the sink belongs to a critical path. However, SERT or SERT-C uses Elmore delay[8], which is not accurate enough in a certain situation [9].

On the other hand, due to the frequent use of automatic logic synthesis tools, in which the longest delay is reduced by changing circuit topology, the number of paths with a similar long delay tends to increase. Hence, we sometimes have a net with more than one critical sink. Therefore, we must take all critical sinks into consideration in the interconnect tree design [7, 10, 11].

The interconnect tree design problem can be formulated in a few ways, such as (i) minimizing the maximum delay of critical sinks[11], (ii) minimizing the weighted sum of delays of critical sinks[10, 12], (iii) minimizing the weighted sum of squared delays of critical sinks, and so on. Each of them has merits and demerits. For example, the delay must be considered on path-based, not on net-based, and hence formulation (i) may not be appropriate, when all the nets in several critical paths are treated. On the other hand, formulation (iii) can treat a criticality of each critical sink and also can reduce the maximum delay, but it requires more computational efforts.

In this paper, we propose two algorithms to find an interconnect tree of a given net, such that a total weighted sum $\tau$ of delays to all sinks of the net is minimum (formulation (ii) stated above). The algorithms are based on the Top-Down Timing Driven Router[12], which constructs an interconnect tree from the source to sinks. The algorithms start from a Steiner tree and repeat a tree transformation as monitoring the change of $\tau$. Although we use Elmore delay to estimate the delay to a sink because of its simplicity, it can be replaced by a more accurate one such as PRIMO[9]. Moreover, it is possible to modify the algorithms so as to solve a problem formulated by (iii) stated above.

## II. Preliminaries

Given a *source* $s_0$ and *sinks* $s_1, s_2, \cdots s_n$ of a net on a two-dimensional surface, we want to find a rectilinear tree connecting these source and sinks with the use of horizontal and vertical wire segments. For each sink $s_i$, a *weight* $w_i$, such that $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$, is also given, which indicates the *criticality* of delay $\tau_i$ to

$s_i$. The interconnect tree $T$ to be found is the one with the minimum total weighted delay $\tau$ of all sinks, which is defined by $\tau = \sum_{i=1}^{n} w_i \tau_i$.

In order to estimate the delay $\tau_i$ from the source to sink $s_i$, we assume *driver output resistance* $R_d$ at the source $s_0$, and *load capacitance* $C_j$ at each sink $s_j$. Each wire segment of an interconnect tree is modeled by the lumped RC $\pi$ model, such that the resistance is $r\ell$ and both capacitances are $c\ell/2$, where $r$ and $c$ are the unit wire resistance and capacitance, respectively, and $\ell$ is the length of the wire segment. We assume that $c$ and $r$ are constant on the two-dimensional surface. Then, for a given interconnect tree, we have an RC tree circuit, in which we can calculate the delay $\tau_i$ to sink $s_i$ by a circuit simulator such as SPICE. However, using SPICE is time consuming, and hence a simpler delay estimation technique is necessary.

Elmore delay [8] is such a technique using the first order moment of impulse response. Regarding the source as the *root*, the sinks and Steiner points as *nodes*, and wire segments as *edges*, an interconnect tree is considered as a *rooted tree*. Therefore, we can define the so-called *parent-child* relation between end-nodes of an edge. In such a rooted tree, Elmore delay $\tau_i$ in a tree $T$ can be expressed as $\tau_i = \sum_{v \in P_i} r_v C(T_v)$, where $P_i$ is a set of nodes on the path from the source to sink $s_i$ in $T$, $T_v$ is a subtree of $T$ rooted at node $v$, $C(T_v)$ is the total capacitance contained in $T_v$, and if $v$ is the source, $r_v = R_d$, otherwise, $r_v$ is the resistance of the wire between node $v$ and the parent of $v$. The total weighted sum of Elmore delay $\tau(T) = \sum_{i=1}^{n} w_i \tau_i$ with respect to a tree $T$ is divided into five terms as follows:

$$\begin{aligned} \tau(T) &= R_d \sum_{i=1}^{n} C_i + cR_d\ell(T) + r\sum_k \ell_k \sum_{s_i \in T_k} C_i \sum_{s_j \in T_k} w_j \\ &\quad + \frac{rc}{2}\sum_k \ell_k^2 \sum_{s_j \in T_k} w_j + rc\sum_k \ell_k \ell(T_k) \sum_{s_j \in T_k} w_j \quad (1) \end{aligned}$$

In this equation, $\ell_k$ is the wire length between node $k$ and its parent, and $\ell(T_k)$ is the total length of wires contained in the subtree $T_k$ rooted at node $k$.

The first term of the equation is constant, and the second term is proportional to the total wire length $\ell(T)$ of tree $T$, which can be minimized by constructing a Steiner tree. The third through the fifth terms are related to the unit wire resistance $r$, and according to $r$ becomes large, they become dominant in $\tau(T)$. Therefore, we must find a tree $T$ such that these terms are minimized. Since our problem is equal to Steiner tree problem [3] in the case of $r = 0$, the problem is NP-hard.

Although Elmore delay has a large error in a certain situation[9] and our problem is still NP-hard, using Elmore delay for delay calculation has a few advantages. For example, we can prove that the optimum routing tree minimizing the weighted Elmore delay for a net with two sinks is either Steiner tree or a star such that two sinks are connected directly by shortest paths from the source [12]. Moreover, for any part of an interconnect tree connecting three nodes, either a Steiner tree or a star is optimum, unless the remaining parts of the tree exclusive of the connection of these three nodes are changed. Therefore, Hanan grid[4] can be an appropriate domain to seek an optimum interconnect tree. However, this property may not be hold, if other model such as PRIMO[9] is used for delay estimation.

## III. Algorithm

In this section, we describe the outline of the proposed algorithm for finding an interconnect tree with the minimum weighted delay $\tau(T)$. The algorithm first generates a Steiner tree $T_0$ as an initial temporary tree, and constructs four subtrees by separating $T_0$ at the source (**Step-0**). Then, in **Step-1**, it modifies a temporary tree $T$ by moving a sink from a subtree to another, and finds a temporary tree $T_{best}$ which has a better weighted delay $\tau(T)$. This tree $T_{best}$ indicates a partition of sinks into four sets, and in **Step-2**, the shape of each subtree $T_b$ connecting the sinks of a set is determined. If a subtree $T_b$ contains one or two sinks, or it contains no critical sinks, then the shape of the subtree is the final one. If $T_b$ contains more than two sinks and at least one critical sink, then a point $v$ is determined on the wire segment of $T_b$ connecting to the source. The wire segment from the source to $v$ is a part of the final routing tree, and is not changed anymore. The subtree of $T_b$ rooted at $v$ is regarded as an initial temporary tree, and similar processes to **Step-1** and **Step-2** are executed to this subtree recursively.

**Step-0**[Initialization]:

1. Construct a rectilinear Steiner tree $T_0$ connecting the source and all sinks, and let a temporary tree $T$ be $T_0$.
2. Divide $T$ into initial subtrees $T_b$ $(1 \le b \le 4)$, each of which consists of the source $s_0$, a wire segment connecting to $s_0$, and a subtree connecting below the wire segment. Since at most four wire segments connect to $s_0$, we have four initial subtrees, which may include an empty subtree. (See Fig. 1(a))
3. Empty queue $Q$.

**Step-1**[Subtree-Transformation-4]:

1. Compute $\tau(T)$ with respect to $T$, and set $\tau_{best} := \tau(T)$ and $T_{best} := T$.
2. Modify $T$ by moving sinks one by one from a subtree to a different subtree, so as to minimize $\tau(T)$ with respect to modified $T$. This is done by repeating the following processes **(a)** and **(b)** for each subtree $T_b$ from $b = 1$ to $b = 4$. (See Fig. 1(b))
   - **(a)** While $T_b$ contains a sink, repeat **i** through **iii**.

**i.** For each sink $s$ in $T_b$, compute the decrement of $\tau(T)$, which is attained by moving $s$ to a different subtree $T_{b'}$. There are three possible subtrees to which $s$ is moved. Let $\Delta(s)$ be the largest decrement of $s$ among these three moves. In this movement, $s$ is connected to the nearest point in $T_{b'}$, and $T_b$ is not modified unless $s$ is a leaf in $T_b$.

**ii.** Select a sink $s*$ with the largest decrement $\Delta(s)$ among all sinks, and modify $T$ by moving $s*$ to the subtree $T_{b*}$ such that the movement of $s$ to $T_{b*}$ gives $\Delta(s*)$. (If $\Delta(s*)$ is positive, then $\tau(T)$ is improved.)

**iii.** Compute $\tau(T)$ for this modified $T$, and if $\tau(T)$ is better than $\tau_{best}$, then update $\tau_{best}$ and $T_{best}$ by setting $\tau_{best} := \tau(T)$ and $T_{best} := T$.

**(b)** Set $T := T_{best}$.

**Step-2**[Subtree-Generation]:

1. For each subtrees $T_b$ $(1 \leq b \leq 4)$ of $T$, conduct **(a)**, **(b)**, or **(c)**.

   **(a)** If $T_b$ contains a single sink $s_i$, connect it from the source $s_0$ by a shortest path.

   **(b)** If $T_b$ contains exactly two sinks, construct Steiner tree $T_{b-Steiner}$ and star $T_{b-Star}$ connecting $s_0$ and the sinks of $T_b$, and compute $\tau(T_{b-Steiner})$ and $\tau(T_{b-Star})$. If $\tau(T_{b-Steiner}) \leq \tau(T_{b-Star})$, then select $T_{b-Steiner}$ as the shape of subtree $T_b$. Otherwise, select $T_{b-Star}$.

   **(c)** If $T_b$ contains more than two sinks, construct a Steiner tree connecting all sinks in $T_b$.

2. If the tree $T$ generated by combining the subtrees $T_b$ $(1 \leq b \leq 4)$ constructed in **1** into one, contains more than 4 wire segments connecting to the source $s_0$, then in order that $T$ contains at most 4 wire segments connecting to $s_0$, modify each subtree $T_b$ with a wire segment $W$ connecting to $s_0$ whose direction is not a proper one to $T_b$, where the proper direction of the wire segment connecting to $s_0$ in $T_b$ is the same direction as the wire segment connecting to $s_0$ when $T_b$ is initially generated in **Step-0-2**. This modification is done by shifting wire segment $W$ by one wiring grid from $s_0$ to the proper direction to $T_b$ (see Fig. 2).

3. For each subtree $T_b$ $(1 \leq b \leq 4)$ having more than two sinks and a sink with non-zero weight, conduct **(a)** or **(b)**. The wire segments of a subtree $T_b$ having at most two nodes or no sink with non-zero weight are fixed and form a part of the final interconnect tree.

   **(a)** If $T_b$ contains a Steiner point $s$ separated from $s_0$ by one wiring grid which is generated in **2**, then set the wire segment from $s_0$ to $s$ as the fixed one of the final tree. Then, enqueue pair $(s, T_b)$ into $Q$.

   **(b)** Otherwise, considering the distribution of sinks in $T_b$, determine the position of a point $v$ which becomes the root of a subtree connecting all sinks in $T_b$ (see Fig. 3). And, set the wire segment from $s_0$ to $v$ as the fixed one of the final tree. Then, enqueue pair $(v, T_b)$ into $Q$.

**Step-3**[Top-Down Routing Phase]:

While $Q$ is not empty, execute the following processes.

1. Dequeue $(v, T_b)$ from $Q$, and execute the process **2** and **3** for node $v$ and a subtree $T_b$, which are similar processes to **Step-1** and **Step-2**, respectively.

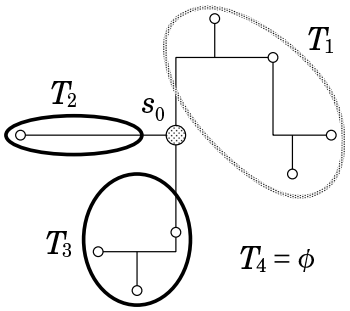
2. [Subtree-Transformation-3]:
   Since at most three wire segments connect to $v$ in $T_b$, divide $T_b$ into three subtrees $T_{b_j}$ $(1 \leq j \leq 3)$, each of which consists of the root $v$ of subtree $T_b$, a wire segment connecting to $v$, and a subtree connecting below the wire segment. Then, modify $T_b$ by moving sinks one by one from a subtree $T_{b_j}$ to a different subtree, so as to minimize $\tau(T_b)$.

3. [Subtree-Generation]:
   For each modified subtree $T_{b_j}$ $(1 \leq j \leq 3)$, determine the fixed part and temporary part, and construct new subtree $T_b$. If a subtree $T_{b_j}$ of modified subtree $T_b$ contains more than two sinks and a sink with non-zero weight, then enqueue pair $(v_j, T_{b_j})$ into $Q$, where $v_j$ is the new root of subtree $T_{b_j}$.

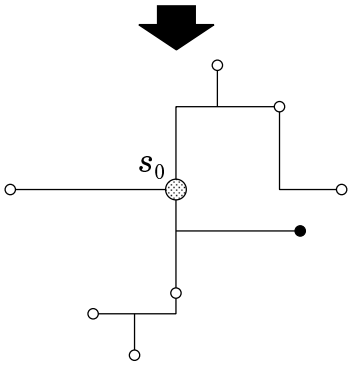
This algorithm is an improved version of $(TD)^2$ router[12]. In $(TD)^2$ router, a Steiner tree is reconstructed for a subtree, every time a sink is moved to the subtree in Subtree-Transformation. Therefore, the time complexity of $(TD)^2$ router is $O(n^2 S(n))$, where $n$ is the number of sinks and $S(n)$ is the time complexity for finding a Steiner tree with $n$ nodes.

In the proposed algorithm, the times needed for **Step-0** and **Step-2** are $O(S(n))$, since finding a Steiner tree is dominant in these steps. In **Step-1**, $O(n)$ time is needed for computing $\Delta(s)$ of each sink $s$, and hence it takes $O(n^2)$ time to find the sink $s*$ with the largest $\Delta(s*)$. Since each sink is not moved more than a constant number, the time complexity of **Step-1** is $O(n^3)$. Therefore, the total time complexity of the proposed algorithm is $O(n^4 + nS(n))$, since the number of repetition of **Step-3** is at most $O(n)$.

Let us call this algorithm TD-old, which is much faster than $(TD)^2$ router, since it does not construct a Steiner tree for every move of a sink in **Step-1-2(a)i**. Moreover,

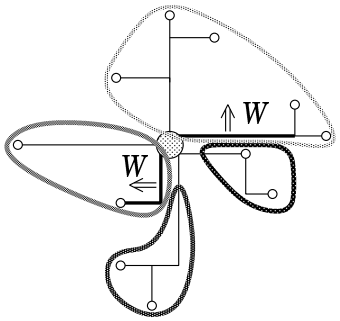



(a) Initial Tree $T_0$ and its subtrees $T_b (1 \leq b \leq 4)$

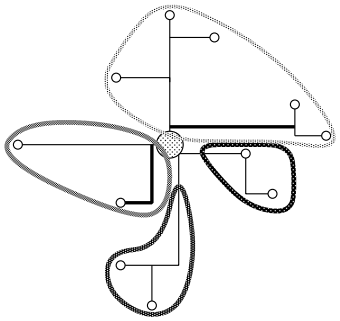



(b) Move of a sink (• moved sink form $T_1$ to $T_3$)

Fig. 1. Subtree-Transformation




(a) More than 4 wire segments connecting to $s_0$



(b) Shift of a wire segment

Fig. 2. Shifting wire segment $W$ by one wiring grid

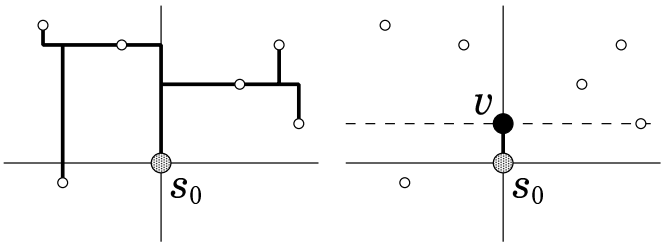



Fig. 3. Fixed wire segment between $s_0$ and $v$

TABLE I
TECHNOLOGIES

| | IC1 (0.5$\mu$m) | IC2 (0.18$\mu$m) | MCM |
|---|---|---|---|
| $R_d$ [$\Omega$] | 270 | 296.5 | 25.0 |
| $r$ [$\Omega/\mu$m] | 0.112 | 0.29 | 0.008 |
| $c$ [fF/$\mu$m] | 0.039 | 0.11 | 0.060 |
| $C_i$ [fF/$\mu$m] | 1.00 | 0.097 | 1000.0 |
| $R_d/r$ [$\times 10^3 \mu$m] | 2.411 | 1.022 | 3.125 |
| Size [mm$^2$] | 2 × 2 | 2 × 2 | 20 × 20 |

TD-old can generate an interconnect tree $T$ with the almost same $\tau(T)$ as (TD)$^2$ router.

In the implementation of the algorithm, we used 1-Steiner algorithm [2] to find a Steiner tree, whose worst case time complexity is $O(n^5)$. Therefore, we need more improvement in the time complexity. In **Step-2-1(c)** of TD-old algorithm, a Steiner tree is sought to reconstruct subtree $T_b$. If we use subtree $T_b$ of $T_{best}$ obtained in **Step-1-2(b)**, then we need not a Steiner tree so that the time complexity of **Step-2** is reduced to $O(n)$. Hence, the total time complexity becomes $O(n^4 + S(n))$. We call this algorithm TD-new in the following.

## IV. Experimental Results

The proposed algorithms are programmed in C language and run on Sun SPARC Station 20/86 to evaluate their performances. In the following, we show a part of the experimental results, which are obtained under three different technologies shown in Table I. In the table, "Size" is the area of the two dimensional surface where the source and sinks are distributed.

We first generated 100 nets randomly, each of which has 8 sinks including one critical sink $s_c$ with $w_c = 1$. The total wire length $\ell(T)$ of the generated tree $T$ and Elmore delay $\tau_c$ to the critical sink $s_c$ are shown in Table II, where each value of the table is an average of 100 data. In the table, "Steiner" and "SERT-C" are the results obtained by 1-Steiner algorithm [2] and SERT-C [13], respectively. Since SERT-C always produced better solutions than Alphabetic tree [10] in our experiments, we omitted the results of Alphabetic tree in the table. Moreover, we omitted the results of (TD)$^2$ router, since the results were

TABLE II
RESULTS (ONE CRITICAL SINK)

| | $\ell(T)$ [mm] | | | $\tau_c$ [ns] | | |
|---|---|---|---|---|---|---|
| | IC1 | IC2 | MCM | IC1 | IC2 | MCM |
| Steiner | 4.32 | 4.32 | 43.2 | 0.0629 | 0.247 | 0.262 |
| SERT-C | 4.43 | 4.52 | 53.5 | 0.0607 | 0.221 | 0.219 |
| TD-old | 4.70 | 5.08 | 53.2 | 0.0599 | 0.201 | 0.219 |
| TD-new | 4.72 | 5.13 | 53.6 | 0.0601 | 0.203 | 0.219 |

TABLE III
COMPARISONS WITH SERT-C AND TD-OLD

| | vs. SERT-C | | | vs. TD-old | | |
|---|---|---|---|---|---|---|
| | win | lose | tie | win | lose | tie |
| IC1 | 55 | 36 | 9 | 16 | 29 | 55 |
| IC2 | 78 | 18 | 4 | 14 | 46 | 40 |
| MCM | 27 | 41 | 32 | 10 | 49 | 41 |

almost same as TD-old.

In Table III, we show the score of the comparison of TD-new with SERT-C and TD-old, where "win" means that $\tau_c$ in the tree obtained by TD-new is better than those obtained by SERT-C and TD-old. From these results we can see that TD-old and TD-new find better trees if resistance ratio becomes small, and TD-new is a little worse than TD-old.

In order to see the effect of the total weighted delay $\tau(T)$, we randomly generated 100 nets, each of which has 8 sinks including two critical sinks. The results are shown in Table IV, where $\tau = 0.5(\tau_1 + \tau_2)$ and $\tau_1$ is the longer delay in the tree obtained by TD-old. Namely, among two critical sinks, the sink which gave the longer delay in the tree obtained by TD-old is named as $s_1$, and the other is named as $s_2$. When SERT-C is applied to these nets, sink $s_1$ is specified as the critical sink. From the table, we can see that in the case of "IC1" our algorithms could not reduce the longest delay, but in the case of "IC2" they could reduce delay $\tau_2$ as well as $\tau_1$. Moreover, as seen from the case of "MCM", delay $\tau_2$ sometimes became larger than $\tau_1$ in the tree obtained by SERT-C.

Table V shows the results for the nets each of which has 16 sinks including 8 critical sinks with the equal weight 1/8, and each value is the average of randomly generated 100 such nets. From the table, we see that TD-old gives better results than TD-new. But, as seen from Table VI, TD-old requires more CPU time than TD-new, where each time is the average of 300 nets. Since SERT-C that we could use cannot treat more than 8 sinks, we cannot show the data.

In our problem, the delay $\tau_i$ to sink $s_i$ and their weighted sum $\tau(T)$ is calculated by Elmore delay. Let us denote by $\tau_{El}(T)$ the weighted sum of Elmore delays to critical sinks in tree $T$. Since Elmore delay has an error, tree $T$ obtained by TD-old or TD-new may be worse than Steiner trees if we calculate delays by SPICE. Namely,

TABLE IV
RESULTS (TWO CRITICAL SINKS)

| | | $\tau_1$ [ns] | $\tau_2$ [ns] | $\tau$ [ns] | $\ell(T)$ [mm] |
|---|---|---|---|---|---|
| IC1 | Steiner | 0.066 | 0.059 | 0.063 | 4.48 |
| | SERT-C | 0.064 | 0.059 | 0.061 | 4.43 |
| | TD-old | 0.065 | 0.058 | 0.061 | 4.44 |
| | TD-new | 0.065 | 0.058 | 0.061 | 4.44 |
| IC2 | Steiner | 0.267 | 0.221 | 0.244 | 4.48 |
| | SERT-C | 0.241 | 0.220 | 0.231 | 4.45 |
| | TD-old | 0.238 | 0.198 | 0.218 | 4.47 |
| | TD-new | 0.241 | 0.199 | 0.220 | 4.47 |
| MCM | Steiner | 0.274 | 0.250 | 0.262 | 44.8 |
| | SERT-C | 0.223 | 0.244 | 0.234 | 45.4 |
| | TD-old | 0.224 | 0.218 | 0.221 | 49.9 |
| | TD-new | 0.224 | 0.218 | 0.221 | 50.6 |

TABLE V
RESULTS (EIGHT CRITICAL SINKS)

| | | $\tau_{max}$ [ns] | $\tau_{min}$ [ns] | $\tau$ [ns] | $\ell(T)$ [mm] |
|---|---|---|---|---|---|
| IC1 | Steiner | 0.108 | 0.078 | 0.096 | 6.20 |
| | TD-old | 0.105 | 0.077 | 0.092 | 6.37 |
| | TD-new | 0.106 | 0.077 | 0.093 | 6.41 |
| IC2 | Steiner | 0.468 | 0.260 | 0.383 | 6.20 |
| | TD-old | 0.417 | 0.257 | 0.330 | 6.97 |
| | TD-new | 0.433 | 0.261 | 0.338 | 7.14 |
| MCM | Steiner | 0.592 | 0.449 | 0.534 | 62.0 |
| | TD-old | 0.436 | 0.409 | 0.421 | 115 |
| | TD-new | 0.444 | 0.423 | 0.434 | 127 |

TABLE VI
CPU TIME [MSEC]

| No. of sinks | 8 | 16 |
|---|---|---|
| Steiner | 118 | 2318 |
| SERT-C | 125 | — |
| TD-new | 134 | 2405 |
| TD-old | 189 | 3639 |

the proposed algorithms may generate trees $T$ such that $\tau_{El}(T) < \tau_{El}(T_0)$ and $\tau_{Sp}(T) > \tau_{Sp}(T_0)$, where $\tau_{Sp}(T)$ is the weighted sum of SPICE delays to critical sinks in tree $T$, and $T_0$ is a Steiner tree. In order to check this, we computed $\tau_{Sp}(T)$ and $\tau_{Sp}(T_0)$ for the nets with 16 sinks including 8 critical sinks.

In the case of "MCM", all the trees $T$ found by TD-new satisfy $\tau_{Sp}(T) \leq \tau_{Sp}(T_0)$, that is, trees with smaller Elmore delay have smaller SPICE delay. However, in the case of "IC1" and "IC2", inversions occurred in 2 nets and 5 nets, respectively. Namely, in these nets, TD-new algorithm found better trees $T$ with respect to the value $\tau_{El}(T)$, but these trees were not better with respect to $\tau_{Sp}(T)$.

PRIMO [9] is a delay calculation technique using the first three moments of circuit response, and more effi-

TABLE VII
RELATIVE ERRORS TO $\tau_{Sp}(T)$

| Tree $T$ | delay | IC1 [%] | IC2 [%] |
|---|---|---|---|
| Steiner | Elmore | +45.6 | +52.0 |
| | PRIMO | −0.39 | −2.71 |
| TD-new | Elmore | +44.8 | +51.6 |
| | PRIMO | −0.56 | +1.78 |

cient than SPICE and more accurate than Elmore. Therefore, it is a good choice for a delay estimation technique. Table VII shows relative errors of $\tau_{El}(T)$ and $\tau_{Pr}(T)$ to $\tau_{Sp}(T)$, where $\tau_{Pr}(T)$ is the weighted sum of PRIMO delays in tree $T$. Each value in the table is an average of randomly generated 100 nets with 16 sinks including 8 critical sinks.

However, even if PRIMO is used, the inversions occurred similarly to the case of Elmore delay, in 2 nets in the case of "IC1" and in 4 nets in the case of "IC2". Namely, for the trees $T$ obtained by TD-new algorithm and Steiner tree $T_0$, we have $\tau_{Pr}(T) < \tau_{Pr}(T_0)$ and $\tau_{Sp}(T) > \tau_{Sp}(T_0)$. These results show that PRIMO is not accurate enough either. Therefore, considering an appropriate formulation of interconnect topology design with the use of an accurate delay model is one of the most important future works.

## V. Conclusion

In this paper, we proposed two algorithms to find an interconnect tree of a net such that the total weighted sum of Elmore delays is minimal. The time complexities of the proposed algorithms, TD-new and TD-old, are $O(n^4 + nS(n))$ and $O(n^4 + S(n))$, respectively, where $n$ is the number of sinks and $S(n)$ is the time complexity for finding a Steiner tree with $n$ nodes. We also showed some experimental results to evaluate the performances of the proposed algorithms. From them, we can see that the algorithms generate better trees than the previous algorithms, and TD-old has a good chance of getting a better tree in a reasonable time, if the number of sinks is small.

There still remain future works. As is mentioned above, an appropriate formulation of the problem with the use of an accurate delay model is important. Moreover, deleting the assumption that the unit wire capacitance is the same on the whole routing area is also important.

## Acknowledgements

The authors express their deep appreciation to Dr. K.D.Boese and Prof. A.B.Kahng, UCLA, and Dr. A.Vittal and Prof. M.Marek-Sadowska, UC Santa Barbara, for their kind offers of their programs of SERT-C and Alphabetic Tree.

## References

[1] J.Cong, Z.Pan, L.He, C.K.Koh, K.Y.Khoo : "Interconnect design for deep submicron ICs", Dig. Tech. Papers ICCAD, pp. 478–485 (1997).

[2] A.B.Khang and G.Robins: "A new class of iterative steiner tree heuristics with good performance", IEEE Trans.CAD, **vol.11**, no.7, pp. 893–902 (1992).

[3] M.R.Garey and D.S.Johnson: "The rectilinear Stener tree problem is NP-complete", SIAM J. Appl. Math., **vol.32**, no.4, pp. 826–834 (1977).

[4] M.Hanan: "On steiner's problem with rectilinear distance", SIAM J. Appl. Math., **vol.11**, no.4, pp. 255–265 (1966).

[5] J.Cong, K.S.Leung, and D.Zhou : "Performance driven interconnect design based on distributed RC delay model", Proc. Design Automation Conf., pp. 606–611 (1993).

[6] H.Mitsubayashi, A.Takahashi, Y.Kajitani : "Cost-radius balanced spaning/steiner trees", Proc. APCCAS96, pp. 377–380 (1996).

[7] K.D.Boese, A.B.Kahng, G.Robins : "Near-optimal critical sink routing tree constructions", IEEE Trans.CAD, **vol.14**, no.12, pp. 1417–1436 (1995).

[8] J.Rubinstein, P.Penfield, M.A.Horowitz : "Signal delay in RCtree networks", IEEE Trans.CAD, **vol.CAD-2**, no.3, pp. 202–211 (1983).

[9] R. Kay and L. Pilleggi: "PRIMO: Probability interpretation of moment for delay calculation", Proc. Design Automation Conf., pp. 463–468 (1998).

[10] A.Vittal and M.Marek-Sadowaska: "Minimal delay interconnect desing using alphabetic trees", Proc. Design Automation Conf., pp. 392–396 (1994).

[11] H. Hou and S. S. Sapatnekar: "Routing tree topology construction to meet interconnect timing constraints", Proc. Int. Symp. Physical Design, pp. 205–210 (1998).

[12] K.Baba, N.Tsujii, K.Yamamoto, and S.Tsukiyama : "A delay minimization router: $(TD)^2$-Router", Proc. APC-CAS98, pp. 117–120 (1998).

[13] K.D.Boese, A.B.Kahng, B.A.McCoy, G.Robins : "Fidelity and near-optimality of Elmore-based routing constructions", Dig. Tech. Papers ICCAD, pp. 81–84 (1993).