An Efficient Hierarchical Timing-Driven Steiner Tree Algorithm for Global Routing*

Jingyu Xu, Xianlong Hong, Tong Jing, Yici Cai
Dept. of Computer Science & Technology, Tsinghua University, Beijing, P. R. China, 100084
Email: xujy, hongxl, jingt, caiyc@tiger.cs.tsinghua.edu.cn
Jun Gu
Dept. of Computer Science, Hong Kong University of Science & Technology, Hong Kong, China
Email: gu@cs.ust.hk

Abstract

In this paper, we propose a hierarchical timing-driven Steiner tree algorithm for global routing which considers the minimization of timing delay during the tree construction as the goal. The algorithm uses heuristic approach to decompose the problem of minimum delay Steiner tree into hierarchy and to construct the sub-trees respectively based on dynamic programming technique. Taking the net topology into consideration, we build the final routing tree by reconnecting the sub-trees at each level recursively and then improve the connection with the objective of minimizing the delay from source to sink pins on the critical path. Meanwhile, some efficient strategies have been proposed to speed up the solving process. Experimental results are given to demonstrate the efficiency of the algorithm.

1. Introduction

With progress in VLSI deep-submicron (DSM) technology, interconnection delay has become increasingly significant in determining circuit speed [1, 2]. The wire capacitance and resistance can no longer be ignored during timing delay calculation since they are comparable to gate capacitance and output-driver resistance [3]. At the global routing stage, which determines how signal nets will finally be connected, it is necessary to estimate the wire delays in order to accurately maximize the overall chip performance.

The Steiner tree algorithm is the essential part of a global routing algorithm. It has been an active field of research. To deal with the interconnection delay, many methods have been proposed, such as wire sizing [4-6], buffer inserting [7-9]. In [10], an interconnection delay calculation technique for custom chip design is presented. However, most existing Steiner tree algorithms of either wire-mode global routers or timing-driven global routers take the minimization of total wire length as the objective. Only limited progress has been reported on timing-driven Steiner tree constructing. [11] proposed a bounded radius Steiner tree global routing algorithm with a wiring cost factor ε. For a small ε the algorithm may result in a very large net wire length, which may increase the timing delay for all sink pins of the net. Based on a simplified Elmore timing model, an iterative Dreyfus-Wagner based (IDW) [12, 13, 15] and a constructive force directed (CFD) Steiner tree algorithm [12, 14, 15] were presented. By improving the equations in Dreyfus-Wagner (DW) algorithm and introducing the timing optimization during the routing process, IDW constructs Steiner trees with high performance. The major weakness of IDW is that it becomes impractical when solving Steiner tree with more vertices (more than 7 vertices) due to the greatly increased run time. CFD runs much faster than IDW does, but the timing performance of the routing tree is decreased. [16] presented a timing-driven Steiner tree algorithm based on Sakurai timing model. Experimental results in [16] show that using Sakurai timing model is more accurate than using Elmore timing model in Steiner tree constructing. However, the run time of the algorithm [16] is still longer.

In this paper, we present a hierarchical timing-driven Steiner tree algorithm based on Sakurai timing model. The algorithm significantly speeds up the process of routing tree constructing with high timing performance. The remainder of this paper is organized as follows. In Section 2, the Sakurai-delay-based timing model for accurately calculating the delay is discussed, the global routing problem is formulated for symbolic analysis, and the Dreyfus-Wagner (DW) algorithm is introduced. The timing-driven Steiner tree algorithm is given in detail in Section 3. Section 4 presents the efficient strategies. Section 5 shows our experimental results. Section 6 is an overall conclusion.

2. Preliminaries

In this section, we will discuss timing models and give the problem formulation. Then, the Dreyfus-Wagner algorithm will be introduced.

*This work was supported in part by the National Grand Fundamental Research 973 Program of China under Grant No. G-1998030411

2.1. Timing model

The Sakurai-delay-based timing model [17], used in this paper instead of Elmore-delay-based timing model [18, 19] and optimized Elmore-delay-based timing model [20], is modeled as a voltage source with the on-resistance of the transistor R_s, distributed RC lines of resistance r_e, capacitance c_e, and loading capacitance C_z. It accords with actual value. The delay T_{DZ} is given by the following expression:

$$T_{DZ} = \beta R_s (c_e + C_z) + \alpha r_e c_e + \beta r_e C_z \quad (1)$$

Where α equals 1.02 and β equals 2.21, which yields 90% of the signal's final delay time.

In the case of multi-terminal net, the delay is formulated as

$$T_D(s) = \beta R_s C_s$$

$$T_D(w) = T_D(v) + \alpha \hat{r}\hat{c} L_{vw}^2 + \beta \hat{r} L_{vw} C_w$$

Where node *v* is the predecessor node of node *w*, L_{vw} is the wire length from *v* to *w*, C_w is the total capacitance of the rest sink pins and wire segments of the net.

2.2. Problem formulation

With the progress in multi-layer routing technology, routing area expands to the whole plane instead of many channels. Let us assume that the chip has been divided into a rectangular array of $N_{row} \times N_{col}$ cells called global routing cells (GRCs). Global routing graph (GRG) is the dual graph of the graph, which is composed of the gridlines and crossings.

Thus, a net can be specified as a set of nodes in GRG. The problem of routing a net in GRG can be described as a Steiner tree problem of specified nodes in the GRG.

2.3. Dreyfus-Wagner algorithm

As an application of dynamic programming technique, the Dreyfus-Wagner algorithm [21] searches for all the possible Steiner trees connecting *T* (representing the set of pins of a signal net) to minimize the total wire length. We use $S_v(K \cup \{v\})$ to denote the Steiner tree of $K \cup \{v\}$, where $K \subseteq T$ and $v \in V - K$, while the definition of $P_v(K \cup \{v\})$ is similar, with one extra constraint that $degree(v) \geq 2$. Let $p(v, w)$ be the shortest path from *v* to *w* in *G*. The following two recursions are used:

$$P_v(K \cup \{v\}) = \min\{S_v(K' \cup \{v\}) + S_v(K - K' \cup \{v\}) \mid K' \subset K \wedge K' \neq \Phi\} \quad (2)$$

$$S_v(K \cup \{v\}) = \min\{\min\{p(v, w) + S_w(K) \mid w \in K\}, \min\{p(v, w) + P_w(K \cup \{w\}) \mid w \notin K\}\} \quad (3)$$

3. The hierarchical timing-driven Steiner tree algorithm

Our timing-driven Steiner tree algorithm will be proposed in this section utilizing dynamic programming technique and hierarchical mechanism. And Sakurai delay model will be analyzed for optimization objectives regarding different cases.

3.1. Formulation derivation for the Sakurai delay model

To determine the optimization objective, we analyze the contribution of each part of Steiner tree to $T_D(t)$. We have

$$\begin{aligned} T_D(t) &= T_D(s) + \alpha \hat{r}\hat{c} \sum_{xy \in path(s,t)} L_{xy}^2 + \beta \hat{r} \sum_{xy \in path(s,t)} L_{xy} C_y \\ &= \beta R_s C_s + \alpha \hat{r}\hat{c} \sum_{xy \in path(s,t)} L_{xy}^2 + \beta \hat{r} \sum_{xy \in path(s,t)} L_{xy} C_y \\ &= [\beta R_s C_{s-w} + \alpha \hat{r}\hat{c} \sum_{xy \in path(s,w)} L_{xy}^2 + \beta \hat{r} \sum_{xy \in path(s,w)} L_{xy} C_{y-w}] \\ &\quad + [\beta (R_S + \hat{r} \sum_{xy \in path(s,w)} L_{xy}) C_W \\ &\quad + \alpha \hat{r}\hat{c} \sum_{xy \in path(w,t)} L_{xy}^2 + \beta \hat{r} \sum_{xy \in path(w,t)} L_{xy} C_y] \end{aligned} \quad (4)$$

where C_{v-w} is the capacitance between node v and w. The first part(in square brackets) has nothing to do with Steiner tree below *w*. In the second part, only the coefficient of C_W depends on Steiner tree above *w*. Let $R_{sw} = R_s + \hat{r} \sum_{xy \in path(s,w)} L_{xy}$, then we have

Theorem 1 For a given critical node *t* below *w*, we minimize (5) when solving minimum delay Steiner tree below *w*. Here *w* is the pseudo-source.

$$T_d(w, t) = \beta R_{sw} C_w + \alpha \hat{r}\hat{c} \sum_{xy \in path(w,t)} L_{xy}^2 + \beta \hat{r} \sum_{xy \in path(w,t)} L_{xy} C_y \quad (5)$$

For $\forall v \notin path(s, t)$, $T_D(t)$ is only affected by C_{s-w} and C_{y-w}. Thus we only need to minimize the total capacitance of Steiner tree below *v*. We have the following theorem.

Theorem 2 For $\forall v \notin path(s, t)$, the objective is minimizing the total wire length of Steiner tree below *v*.

Utilizing dynamic programming technique, we derive the transfer delay function for each critical node. Let $\mathbf{t} = (t_1, t_2, \cdots, t_n)$ be the vector of critical nodes, $S_v(K \cup \{v\}, R_{sv})$ be the minimum delay Steiner tree of $K \cup \{v\}$ regarding R_{sv}. $P_v(K \cup \{v\}, R_{sv})$ is similar to $S_v(K \cup \{v\}, R_{sv})$, with one extra constraint that $degree(v) \geq 2$. And let $T_d(S_v, t_i)$ be the time delay from *v* to critical node t_i in Steiner tree S_v and $T_d(S_v, \mathbf{t})$ the delay vector of all critical nodes. We have

Theorem 3 The transfer delay functions for minimum delay Steiner tree are

$$T_d(P_v(K \cup \{v\}, R_{sv}), \mathbf{t})$$

$$= \min\{T_d(S_v(K' \cup \{v\}, R_{sv}), \mathbf{t}) + \beta R_{sv} C_{K'-\{v\}} \mathbf{I}_{K'} + T_d(S_v(K - K' \cup \{v\}, R_{sv}), \mathbf{t}) + \beta R_{sv} C_{K-K'-\{v\}} \mathbf{I}_{K-K'}\} \quad (6)$$

$$T_d(S_v(K \cup \{v\}, R_{sv}), \mathbf{t}) = \min\{\min\{T_d(S_w(K, R_{sv} + \hat{r}L_{vw}), \mathbf{t}) + (\beta \hat{r} L_{vw} C_w + \alpha \hat{r} \hat{c} L_{vw}^2)\mathbf{I}_k \mid w \in K\}, \min\{T_d(P_w(K \cup \{w\}, R_{sv} + \hat{r}L_{vw}), \mathbf{t}) + (\beta \hat{r} L_{vw} C_w + \alpha \hat{r} \hat{c} L_{vw}^2)\mathbf{I}_k \mid w \notin K\} \quad (7)$$

where $\mathbf{I}_k = (i_{1K}, i_{2k}, \cdots i_{nK}), i_{jK} = \begin{cases} 1, when\ t_j \in K \\ 0, when\ t_j \notin K \end{cases}$

The above theorems give rise to the following corollary.

Corollary 1 Using the above theorems to construct a Steiner tree with the given set of pins, a minimum delay Steiner tree can be formed.

When solving Steiner tree with less than K pins (K is a threshold for hierarchical decomposition which will be discussed later), we use the above theorems to compute the delay of all possible trees connecting the given K pins from bottom to top recursively, and rebuild the Steiner tree yielding the minimum solution from top to bottom.

3.2. The hierarchical constructing mechanism

Due to the exponential nature of the complexity, it is impractical for the dynamic programming algorithm to process nets with large number of pins under current computing environment. Therefore, the hierarchical mechanism is employed to speed up the construction process. At each level, the block of pins is decomposed to L sub-blocks according to their locations. Our two-step approach for decomposition is

Step.1 Assign the Key Nodes of sub-blocks $R_1, R_2, \cdots, R_L$

Search for two vertices r_1, r_2 with the maximal distance and assign them to be the Key Node of R_1 and R_2 respectively. Then for $\forall r \in R$, compute $d(r) = \sum p(r, R_i)^{-1}$. Let vertex r_3 yielding the minimum $d(r)$ be the Key Node of R_3. Repeat this process until we find the Key node of R_L, or until $d(r)$ reaches the threshold.

Step.2 Expand Key Node $r_1, r_2, \cdots, r_L$ to $R_1, R_2, \cdots, R_L$

For $\forall r \in R / \bigcup R_i$, compute $p(r, R_i)$, and insert vertex r yielding the minimum $p(r, R_i)$ into R_i. Repeat this process until $R = \bigcup R_i$.

The estimated complexity of our hierarchical decomposition approach is $O(|R|^2)$.

After constructing the minimum delay Steiner tree for each sub-block respectively, we reconnect these sub-trees by vertices called Surface Node, which are appointed taking into account of the shape of the whole tree. Each sub-tree is regarded as a single "node" at first. Their locations are represented by their own center of gravity. We connect these "nodes" by minimizing the delay from source node s to critical node t. Then sub-trees are unfolded and Surface Nodes are used to connect them instead of the center of gravity. This procedure is recursively processed on each level till the final routing tree is built up. An example is shown in Figure 1.

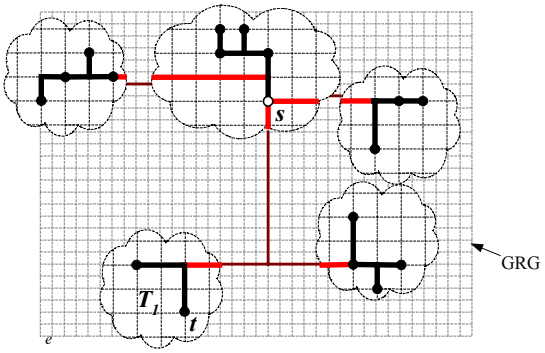

Figure 1. **Decomposition and reconnect of sub-trees**

Note that the construction of each sub-tree should not be isolated from that of other sub-trees. Otherwise the shape of the final tree will be out of control. To make each sub-tree concentrate around the source node, we use the Force-Node to "pull" all the sub-trees together. In Figure 1, when constructing the sub-tree T_1 containing t, we let s be the Force-Node. S pulls the sub-tree closer, thus we get a shorter distance from s to edges of T_1.

3.3. Algorithm description

The pseudo-code of our algorithm is presented in Figure 2.

```
#define maxSimpleSetSize  K
//K is a threshold for hierarchical decomposition
void SteinerTreeCompond(set, path, forceNode)
{
    if (NumberOfNodes ≤ maxSimpleSetSize)
    {
        BuildSimpleTimeDelaySteinerTree(set, path);
    }
    else
    {
        Assign the Key Nodes of sub-blocks  R1,R2,⋯ , RL ;
        Expand Key Node  r1,r2,⋯ , rL  to  R1,R2,⋯ , RL ;
        Create subSet and subPath of each  Ri ;
        for (each  Ri )
        {
            Assign the Force-Node of  Ri ;
            SteinerTreeCompond(subSet[i], subPath[i], Force-Node);
            Compute the center of gravity of  Ri ;
        }
        SteinerTreeCombine(path, subPath, Force-Node);
    }
}

void BuildSimpleTimeDelaySteinerTree(set, path)
{
```

```
        Create all the complementary sub-set pairs;
        Recursively solve the minimum delay Steiner tree, build stack;
        Back stack, search for Steiner points and sub-set pairs yielding the
            minimum solution, put into sets;
        Reconstruct the Steiner Tree according to sets information;
        Free sets and stacks;
}

void SteinerTreeCombine(path, subPath, forceNode)
{
        Let the center of gravity represent the sub-tree, form a virtual graph;
        Find the minimum delay Steiner tree of the virtual graph;
        Unfold each sub-tree and compute the Surface-Nodes to improve the
            connection;
        Connect sub-trees by Surface-Nodes, meanwhile check cycle and
            break it;
}
```

Figure 2. **Pseudo-code of our algorithm**

4. Some efficient strategies in the algorithm

To speed up the construction process of the minimum delay Steiner tree, some efficient strategies will be proposed in following subsections.

4.1. Fast topology constructing

The topology of Steiner tree regarding $P_v(K)$ or $S_v(K)$ depends on K and v. When taking delay into account, their topologies are also affected by R_{sv} according to (5). Thus we can solve the Steiner tree of $P_v(K)$ or $S_v(K)$ from bottom to top to avoid rebuilding sub-trees with same shape. We have

Theorem 4 Suppose that $R_{sv1} < R_{sv2}$ and their $P_v(K)$ or $S_v(K)$ have the same topology, then for $\forall R_{sv} \in [R_{sv1}, R_{sv2}]$, the topology of $P_v(K)$ or $S_v(K)$ will be the same as that of R_{sv1} and R_{sv2}.

Proof: $\exists \lambda \in [0,1]$ that has $R_{sv} = (1-\lambda)R_{sv1} + \lambda R_{sv2}$. (5) can be transformed into $T_d = aR_{sv} + b$, where parameters a and b depends on the topology of the Steiner tree. Let a_{1s}, b_{1s} and a_{2s}, b_{21s} be the parameters of the minimum delay Steiner tree $S_v(K)$ for R_{sv1} and R_{sv2} respectively, the delay of each case are $T_{d1}(v,t)$ and $T_{d2}(v,t)$. We have already known $a_{1s} = a_{2s}$, $b_{1s} = b_{2s}$, we have $T_{d0}(v,t) = a_{s1}R_{sv} + b_{s1} = (1-\lambda)T_{d1}(v,t) + \lambda T_{d2}(v,t)$

Suppose a Steiner tree satisfies $T_d(v,t) = aR_{sv} + b < T_{d0}(v,t)$. Since $T_{d1}(v,t)$ and $T_{d2}(v,t)$ are minimum delay, we have $aR_{sv1} + b \geq T_{d1}(v,t)$ and $aR_{sv2} + b \geq T_{d2}(v,t)$. Then:

$$\begin{aligned} T_d(v,t) &= aR_{sv} + b = a[(1-\lambda)R_{sv1} + \lambda R_{sv2}] + b \\ &= (1-\lambda)(aR_{sv1} + b) + \lambda(aR_{sv2} + b) \\ &\geq (1-\lambda)T_{d1}(v,t) + \lambda T_{d2}(v,t) = T_{d0}(v,t) \end{aligned}$$

The result conflicts with the hypothesis, for $S_v(K)$, theorem 5 is true. For $P_v(K)$ we can get the same conclusion. The construction procedure of the minimum delay Steiner tree is accelerated by using the above proposition. The detailed approach is:

A stack structure is defined to store K, v, R_{s1}, R_{s2} and corresponding descending method of $P_v(K)$ or $S_v(K)$. The structure builds up a hierarchical result stack according to K and v. Before solving $P_v(K)$ and $S_v(K)$ regarding R_s, we check the stack for whether the current case has already been solved. If so, the solution can be reached directly. For each new solution, we have to search the stack for equivalent solutions and combine them. Each level of the stack is sorted by R_{s1} in order to accelerate the searching and combining procedure. Finally, the Steiner tree is rebuilt from top to bottom according to information stored in stack.

4.2. Overlapped edges

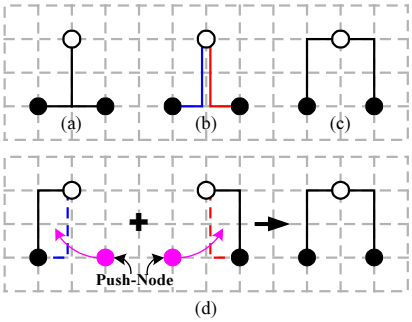

Figure 3. **Overlapped Edges**

When taking timing delay into account, Steiner trees with different topologies may yield the same minimum delay, for which the solutions may include overlapped edges. Figure 3 shows an example of this case. Let us assume that the minimum delay Steiner tree is the hollow node (source) connecting the two solid nodes respectively, which can be represented by either Figure 3(b) or Figure 3(c). Here (b) and (c) have the same delay characteristic. Since overlapped edges are forbidden in GRG, the overlapped edges in (b) will be combined and modified into (a), in which the delay characteristic has changed.

To avoid overlapped edges, we take advantage of the Push-Node. Take Figure 3 as an example. When connecting the hollow node to the left solid node, we let the right solid node be its Push-Node. The Push-Node will "push" the edges connecting left node as far as possible. Also, when connecting the hollow node to the right solid node, we let the left node be its Push-Node. Thus we get the topology in Figure 3(d). In more complex cases, our strategy proved to be efficient in suppressing overlapped edges.

4.3. Cycle-elimination

Unlike the minimum wire length Steiner tree algorithms, cycle is a common problem in timing-driven Steiner tree construction since the final routing tree is

solved from bottom to top. Further, in the hierarchical Steiner tree algorithm, sub-trees need to be connected together based on existing connections, for which cycle may also occurs. Here we propose an efficient strategy to deal with both cases.

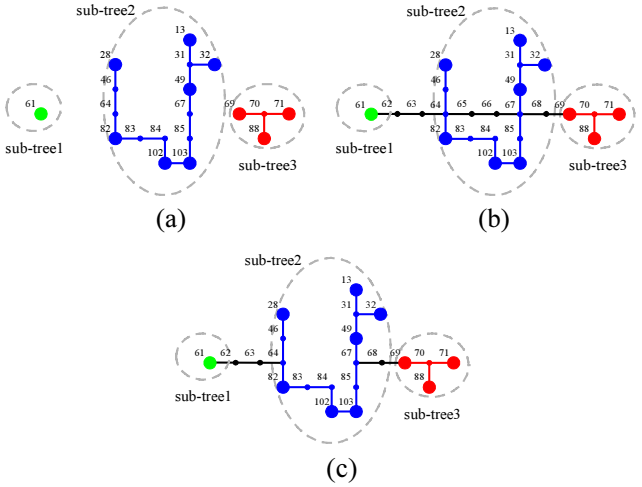

Figure 4. **Cycle in Steiner tree**

We use Figure 4 to explain the case. In Figure 4(a), 3 sub-trees need to be reconnected. The Surface Node of each sub-tree is Node 61, Node 67 and Node 69 respectively. As shown in Figure 4(b), when sub-trees are connected together by Surface-Nodes, the connection of 67 to 61 are represented by path 67-64, and then 64-61, which causes the occurrence of cycle in sub-tree2. A strategy is proposed to eliminate the cycle.

Step 1 Use *c_Path* to denote the path that will connect the Surface-Nodes SN1 and SN2 of sub-tree ST1 and ST2 respectively.

Step 2 $SN1 \rightarrow v1$, $SN2 \rightarrow v2$

Step 3 Search the point p on the *c_Path* from SN1 to SN2, if $p \in ST1$, $p \rightarrow v1$, if $p \in ST2$, stop the search.

Step 4 Search the point q on the *c_Path* from SN2 to p, if $q \in ST2$ $q \rightarrow v2$, if $q = p$, stop the search.

Step 5 Connect $v1$ and $v2$, stop.

The result of our approach is shown in Figure 4(c).

5. Experimental results

The timing-driven Steiner tree algorithm has been implemented in C language on Sun Enterprise 450 under Unix. The experimental results are compared with those of the algorithms mentioned above [12-16].

Microelectronics Center of North Carolina (MCNC) benchmark circuits are used as the test data. According to (1), let $\alpha = 1.02$ and $\beta = 2.21$, T_{DZ} yields 90% of the signal's final delay time [17]. We transform parameters under $2\mu m$ technology to the technology of $0.2\mu m$. According to QCE rule [22], We select $\delta = 10$, $\varepsilon = \alpha^{0.8}$.

Note that the choosing of K will affect the performance of the algorithm. Experimental results indicate that the delay performance will be slightly improved when K increases. As a counterpoint for both performance and run time, K is fixed to be 5 pins in implementation.

5.1. Comparison on run time and total delay

Table 1 compares the results of our approach with the optimal solutions in [12-16]. Row index is the circuit name and serial number of nets in MCNC benchmark. Comparisons of delay performance and run time are shown in column 4 to 6, column 7 to 8 respectively. We can see from the table that the hierarchical algorithm achieves the almost optimal solution while greatly speed up the construction process of minimum time delay Steiner tree. Comparison of run time is plotted in Figure 5, where each dot represents the average run time of nets with same number of pins. With the increase in net size, the speed-up trend becomes more significant that for most nets the speedup is in the order of 1000x ~ 100000x with no degradation of the delay performance.

5.2. Comparison on wire length

Table 2 gives the comparison of the wire length characteristics of our approach with the minimum wire length Steiner tree algorithm. For most of the test nets, the total wire length of our approach is controlled within 5% above the minimum wire length.

6. Conclusions

We have presented an efficient timing-driven Steiner tree algorithm for global routing based on Sakurai delay model. By decomposing the minimum time delay Steiner tree problem into hierarchy, the high-quality solutions are provided with a significant speed up. Our approach has been tested with MCNC benchmark circuits for time delay and total wire length performances. Comparison on these characteristics show that an intelligent heuristic will produce quality equivalent to the direct dynamic programming algorithms and a total speed up of 1000x ~ 100000x due to hierarchical decomposition of the whole problem. For nets with large number of pins, our approach also achieves satisfactory results in a very short time.

References

[1] X. L. Hong, X. L. Yan, C. G. Qiao, *The Theories and Algorithms for VLSI Layout Design*, Beijing: Science Press. 1998.

[2] T. Jing, X. L. Hong, Y. C. Cai, et al, "The Key Technologies and Related Research Work of Performance-Driven Global Routing", *J. of Software*, 2001, 12(5), pp. 677-688.

[3] X. L. Hong, T. X. Xue, E. S. Kuh, et al, "Performance-Driven Steiner Tree Algorithms for Global Routing", *Proceedings of 30th Design Automation Conference (DAC)*, Dallas, Texas, 1993, pp. 177-181.

[4] Jason Jingsheng Cong, Kwok-Shing Leung, "Optimal Wiresizing under Elmore Delay Model", *IEEE Trans. on CAD*, 1995, 14(3), pp. 321-336.

[5] Jason Cong, Lei He, "Optimal Wiresizing for Interconnects with Multiple Sources", *ACM Trans. on Design Automation*

of Electronic Systems. 1996, 1(1-4), pp. 478-511.

[6] John Lillis, Chung-Kuan Cheng, Ting-Ting Y. Lin et al, "New Performance Driven Routing Techniques with Explicit Area/Delay Tradeoff and Simultaneous Wire Sizing", *Proceedings of 33rd Design Automation Conference (DAC)*, Las Vegas, Nevada, 1996.

[7] Chris C. N. Chu, D. F. Wong, "An Efficient and Optimal Algorithm for Simultaneous Buffer and Wire Sizing", *IEEE Trans. on CAD*, 1999, 18(9), pp. 1297-1304.

[8] Jiang Hu, Sachin S. Sapatnekar, "Algorithm for Non-Hanan-Based Optimization for VLSI Interconnect under Higher-Order AWE Model", *IEEE Trans on CAD*, 19(4), 2000, pp. 446-458.

[9] John Lillis, Chung-Kuan Cheng, "Timing Optimization for Multisource Nets: Characterization and Optimal Repeater Insertion", *IEEE Trans on CAD*, 18(3), 1999, pp. 322-331.

[10] Somchai Prasitjutrakul, William J. Kubitz, "A Timing-Driven Global Router for Custom Chip Design", *ICCAD'90*, 1990, pp. 48-51.

[11] Jason Cong, A. Kahng, G. Robins et al, "Provably Good Performance-Driven Global Routing", *IEEE Trans on CAD*, 10(2), 1991, pp. 120-129.

[12] X. L. Hong, T. X. Xue, C. K. Cheng et al, "Performance-Driven Steiner Tree Algorithm for Global Routing", *Proceedings of 30th Design Automation Conference (DAC)*, Dallas, Texas, 1993, pp. 177-181.

[13] X. L. Hong, "A Performance-Driven Steiner Tree Algorithm for Global Routing", *Chinese J. Computers*, 1995, 18(4), pp. 266-272.

[14] X. L. Hong, "A Performance-Driven Steiner Tree Algorithm Using Constructed Force Directed Approach for Global Routing", *Chinese Journal of Semiconductors*, 1995, 16(3), pp. 218-223.

[15] X. L. Hong, T. X. Xue, E. S. kuh, C. K. Cheng, J. Huang, "TIGER: An Efficient Timing-Driven Global Router for Gate Array and Standard Cell Layout Design", *IEEE Trans. on CAD*, 1997, 16(11), pp. 1323-1330.

[16] H. Y. Bao, X. L. Hong, Y. C. Cai, "Timing-Driven Steiner Tree Algorithm Based on Sakurai Model. Chinese Journal of Semiconductors", 1999, 20(1), pp. 41-46.

[17] T. Sacurai, "Approxiamation of Wiring Delay in MOSFET LSI", *IEEE Journal of Solid-State Circuits (SSC)*, 1983, 18(4), pp. 418-426.

[18] W. C. Elmore, "The Transient Response of Lumped Linear Networks with Particular Regard to Wideband Amplifiers", *Journal of Applied Physics*, 1948, 19(1), pp. 55-59.

[19] Rohini Gupta, Byron Krauter, Bogdan Tutuianu et al, "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals", *Proceedings of 32nd Design Automation Conference (DAC)*, San Francisco, CA, 1995, pp. 364-369.

[20] Chung-Ping Chen, Yao-Ping Chen, D. F. Wong, "Optimal Wire-Sizing Formula Under the Elmore Delay Model", *Proceedings of 33rd Design Automation Conference (DAC)*, Las Vegas, Nevada, 1996, Session 26.4.

[21] S. E. Dreyfus, R. A. Wagner, "The Steiner Problem in Graph", *Networks*, 1972, 1, pp. 195-207.

[22] X. W. Gan, *The ABC of Digital CMOS VLSI Analysis and Design*. Beijing, Beijing Univ. Press. 1999.

Table 1. **Comparison on total delay and run time**

Net:No.	Number of Pins	Optimal Delay (ns)	Hierarchical Delay(ns)	% Delay Increase	Optimal Solution CPU Time(ms)	Hierarchical CPU Time(ms)
C5:202	6	0.0255	0.0255	0.00%	70	2.65
C5:408	6	0.0108	0.0108	0.00%	20	2.37
C5:637	6	0.0318	0.0330	3.77%	150	6.49
C5:700	6	0.0388	0.0399	2.84%	70	7.64
C5:706	6	0.0315	0.0318	0.95%	70	7.81
C7:201	6	0.0278	0.0278	0.00%	30	2.84
C7:301	6	0.0279	0.0284	1.79%	60	6.22
C7:611	6	0.0228	0.0239	4.82%	40	2.91
C5:33	7	0.0427	0.0433	1.41%	160	5.56
C5:181	7	0.0228	0.0228	0.00%	170	2.95
C5:226	7	0.0266	0.0273	2.63%	310	2.96
C5:537	7	0.0791	0.0822	3.92%	410	4.17
C7:613	7	0.0283	0.0288	1.77%	350	5.23
C7:221	7	0.0219	0.0220	0.46%	270	6.14
C5:194	8	0.0328	0.0328	0.00%	770	5.74
C5:216	8	0.0253	0.0253	0.00%	2490	5.97
C7:343	8	0.0354	0.0388	9.60%	1570	13.25
C7:344	8	0.0328	0.0328	0.00%	3230	15.91
C5:257	9	0.0189	0.0189	0.00%	1890	3.46
C5:362	9	0.0235	0.0235	0.00%	8130	8.88
C5:541	9	0.1277	0.1338	4.78%	12980	7.10
C5:543	9	0.1130	0.1130	0.00%	21800	9.79
C5:812	9	0.0311	0.0311	0.00%	20790	13.27
C7:337	9	0.0390	0.0394	1.03%	8520	7.30
C5:265	10	0.0221	0.0227	2.71%	10620	8.26
C5:661	10	0.0449	0.0462	2.90%	52340	20.29
C5:840	10	0.0284	0.0284	0.00%	16730	10.90
C5:847	10	0.0288	0.0288	0.00%	30410	5.79
C5:861	10	0.0387	0.0400	3.36%	94090	10.17
C5:387	11	0.0231	0.0240	3.90%	31980	4.39
C5:818	11	0.0374	0.0374	0.00%	447610	8.82
C7:311	11	0.0493	0.0516	4.67%	450600	15.55
C5:86	12	0.0331	0.0343	3.63%	354400	15.14
C5:340	12	0.0871	0.0891	2.30%	1945400	12.14
C7:326	12	0.0348	0.0374	7.47%	336300	9.83

Table 2. **Comparison on wire length**

Net:No.	Number of Pins	Wire Length of Hierarchical Algorithm	Wire Length of Optimal Solution	% Wire Length Increase
C5:265	10	2176	2130	2.16%
C5:340	12	5482	5322	3.01%
C5:387	11	2578	2578	0.00%
C5:537	7	5116	4910	4.20%
C5:541	9	6834	6614	3.33%
C5:637	6	2874	2850	0.84%
C5:661	10	4846	4566	6.13%
C5:700	6	2596	2520	3.02%
C5:847	10	3356	3356	0.00%
C5:861	10	4714	4490	4.99%
C7:337	9	3738	3632	2.92%
C7:301	6	3660	3488	4.93%
C7:311	11	6250	5654	10.54%
C7:344	8	3966	3966	0.00%
C7:611	6	2614	2562	2.03%

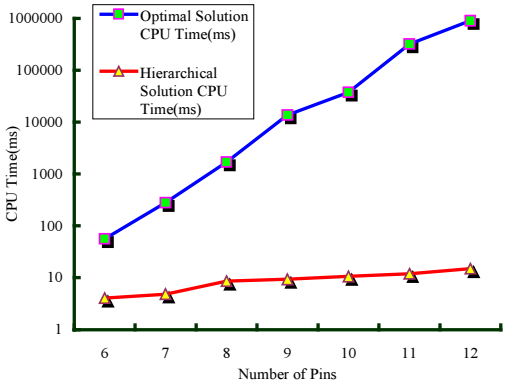

Figure 5. **Comparison of Run Time**