Delay Bounded Buffered Tree Construction for Timing Driven Floorplanning

Maggie Kang Wayne W.-M. Dai
Computer Engineering Department
University of California, Santa Cruz, CA 95064

Tom Dillinger
Rockwell Semiconductor
San Diego, CA 92121

David LaPotin
IBM Austin Research Lab.
Austin, Texas 78758

Abstract

As devices and lines shrink into the deep submicron range, the propagation delay of signals can be effectively improved by repowering the signals using intermediate buffers placed within the routing trees. Almost no existing timing driven floorplanning and placement approaches consider the option of buffer insertion. As such, they may exclude solutions, particularly early in the design process, with smaller overall area and better routability. In this paper, we propose a new methodology in which buffered trees are used to estimate wire delay during floorplanning. Instead of treating delay as one of the objectives, as done by the majority of previous work, we formulate the problem in terms of Delay Bounded Buffered Trees (DBB-tree) and propose an efficient algorithm to construct a DBB spanning tree for use during floorplanning. Experimental results show that the algorithm is very effective. Using buffer insertion at the floorplanning stage yields significantly better solutions in terms of both chip area and total wire length.

1 Introduction

In high speed design, long on-chip interconnects can be modeled as distributed delay lines, where the delay of the lines can often be reduced by buffer insertion. Intermediate buffers can effectively decouple a large load off of a critical path or divide a long wire into smaller segments, each of which has less line resistance and makes the path delay more linear with overall length. It is commonplace for production chips to contain tens of thousands of buffers.

Floorplanning has a significant impact on critical path delay, however almost no existing timing driven floorplanning technique considers the option of buffer insertion. Typically, only wire length or Elmore delay is used for delay calculation. This practice is too restrictive as evidenced by the reliance industry has placed on intermediate buffering as a means for achieving aggressive cycle times.

This paper attempts to leverage the additional freedom gained by incorporating buffer insertion into floorplanning stage. We propose a new methodology of floorplanning using buffered trees to estimate the wiring delay. We formulate the Delay Bounded Buffered Tree (DBB-tree) problem which optimizes the total wire length subject to given timing constraints. Based on the Elmore delay model, we present an efficient algorithm to construct DBB spanning trees for use in floorplanning. The experimental results show that using buffer insertion at the floorplanning stage provides an additional degree of freedom not present in past approaches and typically leads to solutions with significantly smaller area and increased routability.

2 Overview of Related Works

2.1 Elmore Delay Model

The *Elmore delay model* provides a simple closed-form expression with greatly improved accuracy for delay compared to the lumped RC model. For each wire segment modeled as a $\pi-$type circuit, given the interconnect tree T, the Elmore delay from the source s_0 to sink s_i can be expressed as follows:

$$\tau(0,i) = R_0 C_0 + \sum_{e(u,v)\in Path(0,i)} rl_{u,v}\left(\frac{cl_{u,v}}{2} + C_v\right)$$

where R_0 is the driver resistance at the source and C_0 is the total capacitance charged by the driver. Given a uniform wire width, r and c are the unit resistance and capacitance, respectively. Let $e(u,v)$ denotes the wire connecting s_v to its parent s_u, both wire resistance $rl_{u,v}$ and wire capacitance $cl_{u,v}$ are proportional to the wire length $l_{u,v}$. C_v denotes the total capacitance of a subtree rooted at s_v, which is charged through wire $e(u,v)$. The first term of $\tau(0,i)$ is linear with the total wire length of T, while the second term has quadratic dependence on the length of the path from the source to s_i, denoted by $Path(0,i)$.

2.2 Interconnect Optimization

From the discussion of Elmore delay, we can conclude that total tree length and the path length from the driver to the critical sinks are two major concerns for interconnect topology optimization. The early works [1, 2] observed the existence of conflicting min-cost and min-radius (the longest source-to-sink path length of the tree) objectives. A number of algorithms [3, 2, 4, 5, 6]have been proposed to make the trade-offs between them. On the other hand, for deep submicron design, path length is no longer an accurate estimate of path delay. Several attempts [7, 8, 9, 10] have been made to directly optimize Elmore delay rather than using geometric objectives.

Intermediate buffer insertion creates another degree of freedom for interconnect optimization. Early works on fanout optimization problem focused on the construction of buffered trees during logic synthesis [11, 12, 13] without taking into account the wiring effect. Recently, layout driven fanout optimization have been proposed [14, 15]. For a given Steiner tree, a polynomial time dynamic programming algorithm was proposed for the delay-optimal buffer insertion problem [16]. Using dynamic programming, [17] integrated wire sizing and power minimization with the tree construction under a more accurate delay model taking signal slew into account. Inspired by the same dynamic programming algorithm, Okamoto and Cong [18] proposed a simultaneous Steiner tree construction and buffer insertion algorithm. Later the work was extended to include wire sizing [19]. In the formulation of the problem [18, 19], the main objective is to maximize the required arrival time at the root of the tree, which is defined as the minimum among the differences between the arrival time of the sinks and the delay from the root to the sinks.

To achieve optimal delay, multiple buffers may be necessary for a single edge. An early work [20] developed the optimal solution for the size, number and position of buffers driving a uniform line that minimizes the delay of the line. Given single sized buffer, recent work [21] presented the optimal buffer insertion on a uniform wire based on Elmore delay.

2.3 Delay Minimized vs. Delay Bounded

Since floorplanning and placement are usually iterated with static timing analysis tools, the critical path information is often available and the timing requirement for critical sinks converges as the design and layout progresses. It is sufficient to have bounded delay rather than minimized delay. On the other hand, the minimization of total wire length is of interest since total wire length not only contributes to circuit area and routing congestion, but also contributes a significant factor to the switching power. In this paper, instead of minimizing the source to sink delays, we define the delay bounded buffered tree (DBB-tree) problem to minimize the total wire length while satisfying timing constraints. Without considering buffer insertion, [22] proposed a "Delay Bounded Minimum Steiner Tree" (DBMST) to construct a low cost tree subject to bounded delays.

2.4 DBB-tree Problem

In this paper, we define the new Delay Bounded Buffered tree (DBB-tree) problem as follows: Given a signal net and delay bounds associated with critical sinks, construct a routing tree with intermediate buffers inserted to minimize the total wiring length and the number of buffers while satisfying the delay bounds. Based on Elmore delay, we develop an efficient algorithm for constructing DBB spanning tree, which consists of three phases:

1. Calculate the minimum Elmore delay for each critical sink and exclude the floorplanning solution which is timing infeasible.
2. Construct a buffered spanning tree which has minimized total wire length and bounded delays at critical sinks.
3. Revisit the topology obtained in 2 and delete unnecessary buffers subject to delay bounds.

Our DBB-tree algorithm makes the following three major contributions:

- Treating delays as constraints rather than formulating it into the optimized objectives.
- Constructing the topology and inserting buffers simultaneously, the algorithm is very effective to minimize both wire length and the number of buffers.
- Allowing multiple buffers to be inserted on each single edge and calculating the precise buffer positions for the optimal solution. In contrast, most previous work assumes at most one buffer is inserted at the fixed location for each single edge.

3 Description of DBB-tree Algorithm

For floorplanning purpose, we assume uniform wire width and consider only non-inverting buffers. Given a signal net $S = \{s_0, s_1, \cdots, s_n\}$, s_0 is the source and $s_1, \cdots, s_n$ the sinks. The geometric location for each terminal of S is determined by floorplanning. Let $\vec{B} = (t_b, r_b, c_b)$ describes the internal delay, resistance and capacitance of the non-inverting buffer, respectively. Before presenting the detailed DBB-tree algorithm, we first state some theoretical results developed by [21] which calculate the number and position of identical buffers placed on a single edge to minimize the edge delay.

Theorem 1 *Given a uniform line $e(0, i)$ connecting sink s_i to source s_0, and identical buffers $\vec{B}$, the*

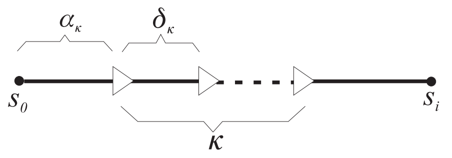

Figure 1: Given a uniform line $e(0,i)$ and identical buffer $\vec{B}$, the Elmore delay through $e(0,i)$ is minimized if $\kappa(0,i)$ buffers are placed on the wire in such way that the first buffer is α_κ away from source s_0, each pair of adjacent buffers are equally spaced each other by δ_κ.

Elmore delay through $e(0,i)$ is minimized if $\kappa(0,i)$ buffers are inserted on the edge such that the first buffer is $\alpha_\kappa(0,i)$ away from the source s_0, and each pair of adjacent buffers are $\delta_\kappa(0,i)$ away from each other :

$$\begin{aligned}
\kappa(0,i) &= \max(0, \lfloor -\frac{1}{2} + \\
&\quad \sqrt{1+\frac{2(r(cl_{0,i}+c_b-c_i)-\ c(r_b-R_0))^2}{rc(r_bc_b+\ t_b)}}\rfloor), \\
\alpha_\kappa(0,i) &= \frac{1}{\kappa+1}\,(l_{0,i}+\frac{t_b(r_b-R_0)}{r}+\frac{c_i-c_b}{c}), \\
\delta_\kappa(0,i) &= \frac{1}{\kappa+1}(l_{0,i}-\frac{r_b-R_0}{r}+\frac{c_i-c_b}{c}).
\end{aligned}$$

where R_0 is the driver resistance at source s_0 and c_i the loading capacitance at sink s_i.

By replacing R_0 with 0, the results can be applied to any wire which connects two sinks in routing tree T. Based on the discussion above, we will present the detailed DBB-tree algorithm in the following section.

3.1 Lower Bound of Elmore Delay

The first phase of DBB-tree algorithm calculates the lower bound of Elmore delay for each critical sink s_i. It may not be possible to achieve this delay simultaneously for all critical sinks, but no achievable delay will be less than it. Formally, the lower bound of Elmore delay for s_i can be given by:

$$\tau^*(0,i) = \tau_\kappa(0,i)$$

where $\tau_\kappa(0,i)$ is the minimized Elmore delay through edge $e(0,i)$ as shown in Figure 1. That is, the lower bound of sink s_i is the minimized delay through the direct connection between source s_0 and s_i without considering other terminals' effect. The floorplanning is timing infeasible if there exists a critical sink s_i such that its lower bound $\tau^*(0,i)$ is greater than the given delay bound D_i : $\tau^*(0,i) > D_i$. The infeasible solution will be excluded, otherwise the algorithm continues to phase 2 and 3.

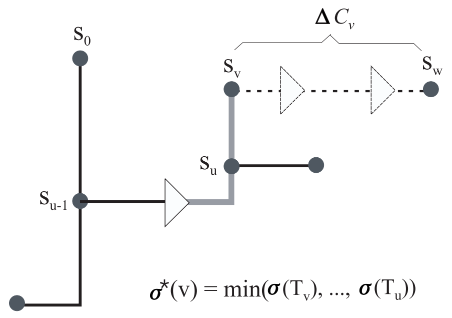

Figure 2: For particular sink $s_v \in T$, the last buffer on edge $e(u-1,u)$ drives T_v, the subtree rooted at s_v, if $e(u-1,u)$ is the last buffered edge on the path between the source and s_v.

3.2 DBB Spanning Tree Construction

The second phase constructs a buffered spanning tree such that the total wire length is minimized and bounded delays are satisfied. Similar with Prim's MST algorithm, it starts with the trivial tree: $T = \{s_0\}$, iteratively edge $e(v,w)$ with $\kappa(v,w)$ buffers is added into T, where $s_v \in T$ and $s_w \in S-T$ are chosen such that the length of $e(v,w)$ is minimized and the timing constraints in T are still satisfied. T grows up incrementally until it spans all terminals, or there is no edge that can be added without exceeding the delay bounds. In the latter case, the floorplanning is considered timing infeasible and the solution is excluded.

For the incremental construction of the DBB-tree, the key issue is how to quickly evaluate the timing constraints each time a new edge is added, i.e. whether or not the delay bound at each critical sink is satisfied. Recall for edge $e(v,w)$, the number of buffers, $\kappa(v,w)$, and the position of buffers, $\alpha_\kappa(v,w)$ and $\delta_\kappa(v,w)$, inserted on the edge which minimize the edge delay can be calculated in constant time. Let T_v denote the subtree rooted at s_v, after adding edge $e(v,w)$ in which $s_v \in T$ and $s_w \in S-T$, the loading capacitance of T_v will be increased by ΔC_v:

$$\Delta C_v = \begin{cases} cl_{v,w}+c_w & \text{if } \kappa(v,w)=0, \\ c\alpha_\kappa(v,w)+c_b & \text{otherwise.} \end{cases}$$

where $l_{v,w}$ is the length of edge $e(v,w)$ and c_w the loading capacitance at sink s_w. Let $e(u-1,u)$ denote the last buffered edge on the path from the source to s_v as shown in Fig. 2, the last buffer on edge $e(u-1,u)$ drives T_v. If there is no buffer from the source to s_v, the source drives T_v directly. According to Elmore delay, T_v is driven through the resistance between the driver and s_v, defined as *driving resistance* of T_v, denoted by $R(T_v)$. Given s_{v-1} is the parent of s_v, and κ, α_κ and δ_κ are abbreviations of $\kappa(v-1,v)$, $\alpha_\kappa(v-1,v)$ and $\delta_\kappa(v-1,v)$, respectively, $R(T_v)$ can be calculated as follows:

$$R(T_v) = \begin{cases} R(T_{v-1}) + rl_{v-1,v} & \text{if } \kappa = 0, \\ r_b + r(l_{v-1,v} - \alpha_\kappa - (\kappa-1)\delta_\kappa) & \text{otherwise.} \end{cases}$$

Let $T_i - T_{i+1}$ denote the set of sinks in subtree T_i but not in T_{i+1} and $s_{u+1}, \cdots, s_{v-1}$ denote the intermediate terminals from s_u to s_v. The Elmore delay of sinks in $T_i - T_{i+1}$ for $i = u, u+1, \cdots, v$, is given by:

$$\Delta\tau(0, s) = R(T_i)\Delta C_v$$

On the other hand, due to the isolation of buffers on edge $e(u-1, u)$, the increased loading capacitance of T_v will not affect on the delay of sinks which are not in T_u. We define the *delay slack* of each sink $s \in T$ as:

$$\lambda(s) = D_s - \tau(0, s),$$

and the delay slack of each subtree T_i as:

$$\lambda(T_i) = \min_{s \in T_i} \lambda(s) \tag{1}$$

The timing constraints will be satisfied for the sinks in $T_u - \{s_w\}$ if and only if for $i = u, u+1, \cdots v$, the following condition holds:

$$\lambda(T_i) \geq R(T_i)\Delta C_v \tag{2}$$

By introducing the *loading capacitance slack* of each subtree T_i :

$$\sigma(T_i) = \frac{\lambda(T_i)}{R(T_i)}, \tag{3}$$

and the minimum loading capacitance slack among subtrees T_i for $i = u, u+1, \cdots v$:

$$\sigma^*(v) = \min_{i=u,u+1,\cdots,v} \sigma(T_i),$$

the condition in Eq. 2 can be simply rewritten as:

$$\sigma^*(v) \geq \Delta C_v.$$

By keeping track of $\sigma^*(v)$, this condition can be checked in constant time. The Elmore delay of s_w can also be calculated from the Elmore delay of s_v:

$$\tau(0, w) = \tau(0, v) + R(T_v)\Delta C_v + \tau_\kappa(v, w).$$

From above analysis, we can conclude:

Theorem 2 *Given a routing tree T with bounded delays at critical sinks, after a new edge $e(v, w)$ is added, the necessary and sufficient condition for satisfying the bounded delays in T is:*

$$\sigma^*(v) \geq \Delta C_v \quad \textit{and} \quad D_w \geq \tau(0, w), \tag{4}$$

By keeping track of $\sigma^*(v)$, this condition can be checked in constant time.

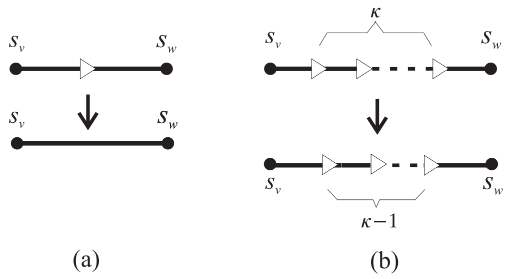

Figure 3: In case of (a), edge $e(v, w)$ becomes unbuffered after deleting the buffer, otherwise $\kappa(v, w) - 1 > 0$ buffers are re-inserted on $e(v, w)$, as shown in (b).

At each iterative step, $s_v \in T$ and $s_w \in S - T$ can be selected in linear time such that $l_{v,w}$ is minimum and the timing constraints are satisfied. After adding the new edge $e(v, w)$, a two-pass traversal of T is sufficient to update the delay slack and loading capacitance slack of each subtree in T: (1) traverse T bottom up and calculate the delay slack and loading capacitance slack of each subtree T_i according to Equations 1 and 3; (2) traverse T top down and calculate $\sigma^*(i)$ from $\sigma^*(i-1)$, given s_{i-1} is the parent of s_i:

$$\sigma^*(i) = \begin{cases} \sigma(i) & \text{if } \kappa(i-1, i) > 0, \\ \min(\sigma^*(i-1),\ \sigma(i)) & \text{otherwise.} \end{cases}$$

Since each new edge can be added into T in linear time, the overall DBB spanning tree can be constructed in $O(n^2)$ time for net S with n terminals.

3.3 Buffer Deletion

In phase 2, buffers are inserted on each edge to minimize the edge delay in order to maximize the choice of tree topology. Some buffers may not be necessary for meeting the delay bounds. The third phase deletes unnecessary buffers from the spanning tree obtained in phase 2. In general, the buffers closest to the source can unload the critical path the most. The algorithm traverses T bottom up and deletes one buffer at a time without violating timing constraints. The deletion continues until all buffers left in T are necessary, that is, the delays would not be bounded if any of the remaining buffers is deleted.

For particular edge $e(v, w)$ with $\kappa(v, w) > 0$ buffers, if one buffer is deleted from $e(v, w)$, the wire delay will be increased by $\Delta\tau_\kappa(v, w)$:

$$\Delta\tau_\kappa(v, w) = \frac{(rcl_{v,w} + r(c_w - c_b) - cr_b)^2}{2\kappa(v, w)(\kappa(v, w) + 1)rc} - t_b - r_b c_b.$$

In case of $\kappa(v, w) = 1$ as shown in Fig. 3 (a), $e(v, w)$ becomes unbuffered edge after deletion, the load capacitance of subtree T_v is increased by:

$$\Delta C_v = cl_{v,w} + C_w - c\alpha_\kappa(v, w) - c_b.$$

Otherwise $\kappa(v,w)-1$ buffers are re-inserted on edge $e(v,w)$, as shown in Fig. 3 (b): $\alpha_\kappa \rightarrow \alpha_{\kappa-1}$ and $\delta_\kappa \rightarrow \delta_{\kappa-1}$. The load capacitance of T_v is increased by :

$$\Delta C_v = c(\alpha_{\kappa-1} - \alpha_\kappa).$$

Similar to phase 2, let $e(u-1,u)$ denote the last buffered edge from the source to s_v. The delay of the sinks in subtree T_u will be increased due to the increased loading capacitance of T_v. In addition, the delay of sinks in subtree T_w will be further increased due to the increased edge delay of $e(v,w)$. Based on the analysis in phase 2, we can conclude that:

Theorem 3 *Given a buffered routing tree T with bounded delays at critical sinks, one buffer can be deleted from edge $e(v,w)$ without causing timing violation if and only if the following condition holds:*

$$\sigma^*(v) \geq \Delta C_v \quad and \quad \lambda(T_w) \geq R(T_v)\Delta C_v + \Delta\tau_\kappa(v,w) \quad (5)$$

Therefore the timing constraints of T can be evaluated in constant time for deleting a buffer from edge $e(v,w)$. The buffer can be found by searching at most $n-1$ edges. After deleting a buffer, the delay slack and loading capacitance slack of subtrees in T are incrementally updated in $O(n)$ time as in phase 2. So one buffer will be deleted in linear time. There are at most kn buffers in T where k is the maximum number of buffers on single edge, The timing complexity of buffer deletion is $O(kn^2)$ which dominates the overall DBB-tree algorithm. Following experimental results show that the buffer deletion effectively minimizes the total number of buffers and it can delete up to 93% of the buffers inserted in the previous phase.

4 Experimental Results

In the first part of the experiments, we implemented the DBB spanning tree algorithm on a Sun SPARC 20 workstation under the C/UNIX environment. The algorithm was tested on signal nets with 2, 5, 10, 25, 50 and 100 pins. For each net size, 100 nets are randomly generated on a $10mm \times 10mm$ routing region, and the delay bounds of critical sinks are randomly chosen from $[1.0ns, 5.0ns]$. The parameters used in the experiments are based on [18], which are summarized in Table 1.

The average results are reported in Table 2, the CPU time consumed per net shows that the algorithm is fast enough that can be applied during the stochastic optimization. The average number of buffers inserted in DBB-tree is very reasonable considering the number of terminals of the net. In addition, the number of buffers before the third phase of DBB-tree algorithm is also listed, the percentage of buffers reduced is as high as 93%. Therefore the buffer deletion is quite effective at removing unnecessary buffers.

Table 1: Experimental Parameters for Signal Nets

Output Resistance of Driver	R_0	500Ω – 1000Ω
Unit Wire Resistance	c	$0.12\Omega/\mu m$
Unit Wire Capacitance	r	$0.15fF/\mu m$
Output Resistance of Buffer	r_b	500Ω
Loading Capacitance of Buffer	c_b	$0.05pF$
Intrinsic Delay of Buffer	t_b	$0.1ns$
Loading Capacitance of Sink	c_i	$0.05pF$ – $0.15pF$

Table 2: Average Results of DBB-trees

Pins	CPU	Number of Buffers		
(#)	(*sec.*)	Before Del.	After Del.	Reduction (%)
2	0.0004	3.24	0.23	92.90
5	0.0019	7.09	1.43	79.83
10	0.0049	13.99	2.82	79.84
25	0.0816	36.34	4.57	87.42
50	0.6259	79.61	7.15	91.02
100	4.9228	171.24	10.53	93.85

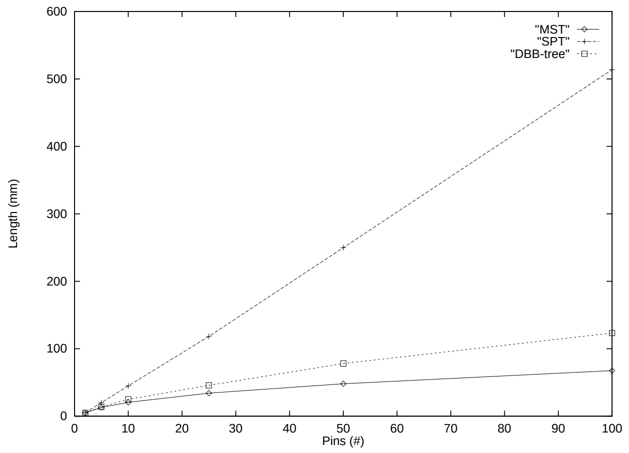

Figure 4: Average length of MST, SPT and DBB-trees.

Furthermore, we compared the average results with the minimum spanning tree (MST) and shortest path tree (SPT) on the same examples. Figure 4 shows the average length achieved by three different trees, DBB-tree is much closer to MST than to SPT.

To compare with the traditional approaches which do not consider buffer insertion during the floorplanning, in the second part of the experiments, we apply DBB-tree, MST and SPT to evaluate the timing performance during the floorplanning, respectively. Four examples are generated, in which blocks, netlists and timing bounds are randomly generated within some nominal ranges. Table 3 reports the consistant improvement of chip area and total wire length by using DBB-tree. The area can be improved up to 31% over MST and 22% over SPT, on the other hand, the total

Table 3: Improvement of Floorplanning Using DBB-tree.

Blocks	Nets	Area(%)		Wire Length(%)	
(#)	(#)	MST	SPT	MST	SPT
10	50	31.10	21.01	14.71	47.04
25	75	27.74	14.28	13.42	43.85
50	150	24.24	9.48	19.23	46.02
100	250	22.27	22.06	14.17	63.04

wire length can be improved up to 19% over MST and 63% over SPT. This substantial improvement demonstrates that using buffer insertion at the floorplanning stage yields significantly better solutions in terms of both chip area and total wire length. It should be noted that future research is needed to extend the approach to distribute buffers into the empty space between macros subject to timing constraints. However, the area of such buffers is typically a small fraction of a given macro area and can be accommodated.

5 Conclusion

In this paper, we propose a new methodology of floorplanning where intermediate buffer insertion is used as another degree of freedom in the delay calculation. A more realistic model for the path based timing driven layout design is defined: Delay Bounded Buffered tree (DBB-tree), in which we treat the delay bounds as constraints rather than formulating the delay into the objectives as is done in most of the previous work. The efficient DBB spanning tree algorithm made our buffered tree based floorplanning very effective and applicable to industrial problems.

References

[1] J. P. Cohoon and L. J. Randall, "Critical net routing," in *Proc. IEEE Intl. Conf. on Computer Design*, pp. 174–177, 1991.

[2] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Performance-Driven global routing for cell based IC's," in *Proc. IEEE Intl. Conf. Computer Design*, (Cambridege, MA), pp. 170–173, October 1991.

[3] J. M. Ho, D. J. Lee, C. H. Chang, and C. K. Wong, "Bounded-diameter spanning tree and related problems," in *Proc. ACM Symp. on Computational Geometry*, pp. 276–282, 1989.

[4] A. Lim and S. Wing Cheng, "Performance oriented rectilinear steiner trees," in *Proc. of 30th Design Automation Conf.*, pp. 171–176, June 1992.

[5] 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," in *Proc. of IEEE Intl. Symp. on Circuits and Systems*, pp. 1869–1872, 1993.

[6] J. Cong, K. Shing Leung, and D. Zhou, "Performance-Driven interconnect design based on distributed RC delay model," in *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 606–611, June 1993.

[7] K. D. Boese, A. B. Kahng, and G. Robins, "High-Performance routing trees with identified critical sinks," in *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 182–187, June 1993.

[8] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Rectilinear steiner trees with minimum elmore delay," in *Proc. 31st ACM/IEEE Design Automation Conf.*, pp. 381–387, June 1994.

[9] X. Hong, T. Xue, E. S. Kuh, C. K. Cheng, and J. Huang, "Performance-Driven steiner tree algorithms for global routing," in *Proc. 30th ACM/IEEE Design Automation Conf.*, (Baltimore, MD), pp. 177–181, June 1993.

[10] S. Prasitjutrakul and W. J. Kubitz, "A timing-Driven global router for custom chip design," in *IEEE Intl. Conf. on Computer Aided Design*, pp. 48–51, 1990.

[11] K. J. Singh and A. Sangiovanni-Vincentelli, "A heuristic algorithm for the fanout problem," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 357–360, 1990.

[12] H. J. Touati, C. W. Moon, R. K. Brayton, and A. Wang, "Performance oriented technology mapping," in *Proc. 6th MIT VLSI Conf.*, pp. 79–97, 1990.

[13] C. L. Berman, J. L. Carter, and K. F. Day, "The fanout problem: From theory to practice," in *Proc. 1989 Decennial Caltech Conf.*, pp. 69–99, 1989.

[14] L. N. Kannan, P. R. Suaris, and H. G. Fang, "A methodology and algorithms for post-Placement delay optimization," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 327–332, 1994.

[15] H. Vaishnav and M. Pedram, "Routability-Driven fanout optimization," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 230–235, 1993.

[16] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal elmore delay," in *Proc. International Symposium on Circuits and Systems*, pp. 865–868, 1990.

[17] J. Lillis, C. Kuan Cheng, and T. Ting Y. Lin, "Optimal and efficient buffer insertion and wire sizing," in *Proc. IEEE 1995 Custom Integrated Circuits Conf.*, pp. 259–262, 1995.

[18] T. Okamoto and J. Cong, "Interconnect layout optimization by simultaneous steiner tree construction and buffer insertion," in *Proc. 5th ACM/SIGDA Physical Design Workshop*, (Reston, Virginia), pp. 1–6, April 1996.

[19] T. Okamoto and J. Cong, "Buffered steiner tree construction with wire sizing for interconnect layout optimization," in *Proc. 1996 IEEE/ACM International Conf. on Computer Aided Design*, (San Jose, CA), pp. 44–49, Nov. 1996.

[20] S. Dhar and M. A. Franklin, "Optimum buffer circuits for driving long uniform lines," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 32–40, January 1991.

[21] C. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *to appear in Proc. of 34th Design Automation Conf.*, June 1997.

[22] Q. Zhu, *Chip and Package Co-Synthesis of Clock Networks*. PhD thesis, Univ. of California, Santa Cruz, Santa Cruz, CA, June 1995.