Zero-Skew Clock Routing Trees With Minimum Wirelength*

Kenneth D. Boese and Andrew B. Kahng

UCLA Computer Science Dept., Los Angeles, CA 90024-1596

Abstract

In the design of high performance VLSI systems, minimization of clock skew is an increasingly important objective. Additionally, wirelength of clock routing trees should be minimized in order to reduce system power requirements and deformation of the clock pulse at the synchronizing elements of the system. In this paper, we present the Deferred-Merge Embedding (DME) algorithm, which in linear time embeds any given connection topology into the Manhattan plane to create a clock tree with zero skew while minimizing total wirelength. Extensive experimental results show that the algorithm yields exact zero skew trees with 9% to 16% wirelength reduction over previous constructions [5] [6]. The DME algorithm may be applied to either the Elmore or the linear delay model, and yields optimal *total wirelength for linear delay.*

1 Introduction

In synchronous VLSI designs, circuit speed is increasingly limited by clock skew, which is the maximum difference in arrival times of the clocking signal at the synchronizing elements. This is seen from the following well-known inequality governing the clock period of a clock signal net [1] [5]:

$$clock\ period\ \geq t_d + t_{skew} + t_{su} + t_{ds}$$

where t_d is the delay on the longest path through combinational logic, t_{skew} is the clock skew, t_{su} is the set up time of the synchronizing elements, and t_{ds} is the propagation delay within the synchronizing elements. With increased switching speeds, t_{skew} may account for over 10% of the system cycle time in high-performance systems [1].

Previous methods for skew minimization [5] [6] [8] concentrate on the problem of computing a clock tree *topology*, and only incompletely address the associated problem of finding a minimum-cost *embedding* of the topology. However, the total wirelength of the clock tree is critical to power consumption and other area/performance parameters of the layout. In this paper, we propose a new approach which achieves exact zero skew while significantly reducing the total wirelength of the clock tree. The basic idea of our Deferred-Merge Embedding (DME) algorithm is to *defer* the embedding of internal nodes in a given topology for as long as possible: (i) a bottom-up phase computes loci of feasible locations for the roots of recursively merged subtrees, and (ii) a top-down phase then resolves the exact embedding of these internal nodes of the clock tree. In practice, the DME algorithm begins with an initial clock tree computed by any previous method, then maintains exact zero clock skew while reducing the wirelength. In regimes where the linear delay model applies, our method produces the *optimal* (i.e., minimum wirelength) zero skew clock tree with respect to the prescribed topology, and this tree will also enjoy *optimal* source-sink delay. Experimental results in Section 6 below show that the DME approach is highly effective in both the Elmore and linear delay models. We achieve average savings in total clock tree wirelength of 16% over the MMM algorithm [5] and 9% over the method of Kahng et al. [6]. In all cases, our clock trees have *exact* zero skew according to the appropriate delay model.

2 Problem Formulation

The placement phase of physical layout determines positions for the synchronizing elements of a circuit, which we call the *sinks* of the clock net. A finite set of sink locations, denoted by $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$, specifies an instance of the clock routing problem. A *connection topology* is defined to be a rooted binary tree, G, which has n leaves corresponding to the set of sinks S. A *clock tree* $T(S)$ is an embedding of the connection topology in the Manhattan plane.[1] In other words, the embedding associates a *placement* in $\Re^2$ with each internal node $v \in G$; we will use $pl(T, v)$ or $pl(v)$ to denote this location. The root of the clock tree is the clock *source*, denoted by s_0. We direct all edges of the clock tree away from the source; a directed edge from v to w may be uniquely identified with w and denoted by e_w. We say that v is the *parent* of w, and w is a *child* of v. The wirelength, or *cost*, of the edge e_w is denoted by $|e_w|$, and must be greater than or equal to the Manhattan distance between its endpoints $pl(w)$ and $pl(v)$.[2] The cost of $T(S)$ is the total wirelength of the edges in $T(S)$.

For a given clock tree $T(S)$, let $t_d(s_0, s_i)$ denote the signal propagation time, or *delay*, on the unique path from source s_0 to sink s_i; the collection of edges in this path is denoted by $path(s_0, s_i)$. The *skew* of $T(S)$ is the maximum value of $|t_d(s_0, s_i) - t_d(s_0, s_j)|$ over all

*This work was supported in part by NSF MIP-9110696, ARO DAAK-70-92-K-0001, and a GTE Graduate Fellowship.

[1] Because the meaning is clear, we use $T(S)$ instead of $T(S, G)$ to denote a clock tree, although implicitly the embedding is always with respect to a particular topology G.

[2] To preserve zero skew, it is sometimes necessary for an edge to have length greater than the distance between its endpoints.

sink pairs $s_i, s_j \in S$. If the skew of $T(S)$ is zero then it is called a *zero skew clock tree* (ZST). Given a set S of sinks, the zero skew clock routing problem is to construct a ZST $T(S)$ of minimum cost. A variant of interest is where the topology is prescribed:

Zero Skew Clock Routing Problem (S,G): *Given a set S of sink locations and a connection topology G, construct a ZST $T(S)$ with topology G and having minimum cost.*

The notion of a zero skew clock tree is well defined only in the context of a method for evaluating signal delays. The delay from the source to any sink depends on the wirelength of the source-sink path, the RC constants of the wire segments in the routing, and the underlying connection topology of the clock tree. In practice simple RC delay approximations, such as the linear model or the Elmore model, are often used to approximate signal delay. Since our construction applies to *any* delay model that is monotone in the wirelength of each edge (e.g., in the linear model, delay is simply given by edge length), we defer details of these delay models to [2][3][8].

3 The Deferred-Merge Embedding (DME) Algorithm

The Deferred-Merge Embedding (DME) algorithm embeds internal nodes of the topology G via a two-phase process. A bottom-up phase constructs a tree of line segments that represent loci of possible placements of the internal nodes in the ZST. A top-down phase then resolves the exact locations of all internal nodes in T. In the discussion that follows, the *distance* between two points p and q is assumed to be the Manhattan distance $d(p,q)$, and the distance between two sets of points P and Q, written $d(P,Q)$, is given by $min\{d(p,q) \mid p \in P \text{ and } q \in Q\}$.

3.1 Phase I: Tree of Merging Segments

For prescribed sink locations S and connection topology G, we construct a tree of *merging segments*. For each node $v \in G$, we construct a merging segment containing a set of possible placements of v. The merging segment of a node depends on the merging segments of its two children, so the topology must be processed in a bottom-up order. In building the tree of merging segments, we also assign a length to each edge in G; this length is retained in the final embedding of G as a ZST.

Let a and b be the children of node v in G. We use TS_a and TS_b to denote the subtrees of merging segments rooted at a and b, respectively. We are interested in placements of v which allow TS_a and TS_b to be merged with *minimum* added wire while preserving zero skew. Define the *merging cost* between TS_a and TS_b to be $|e_a| + |e_b|$, where $|e_a|$ and $|e_b|$ are the lengths to be assigned to edges e_a and e_b. Since delay is a monotone increasing function of wirelength, there is a unique assignment to $|e_a|$ and $|e_b|$ that minimizes merging cost while balancing delays at $pl(v)$.

A *Manhattan arc* is a line segment, possibly of zero length, with slope +1 or -1; in other words, a Manhattan arc is a line segment tilted at 45 degrees from the wiring directions. The collection of points within a fixed distance of a Manhattan arc is called a *tilted rectangular region*, or *TRR*, whose boundary is composed of Manhattan arcs (see Figure 1). The *core* of a TRR is the subset of the TRR at maximum (Manhattan) distance from its boundary; this subset is always a Manhattan arc. The *radius* of a TRR is the distance between its core and its boundary.

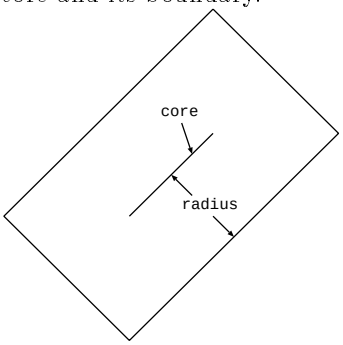

Figure 1: An example of a TRR.

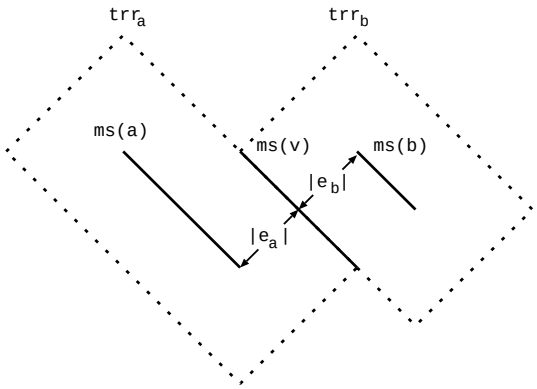

Figure 2: Construction of merging segment $ms(v)$.

The *merging segment* of node v, $ms(v)$, is defined recursively as follows: if v is a sink s_i, then $ms(v) = \{s_i\}$. If v is an internal node, then $ms(v)$ is the set of all points within distance $|e_a|$ of $ms(a)$ and within distance $|e_b|$ of $ms(b)$. If $ms(a)$ and $ms(b)$ are both Manhattan arcs, then we obtain the merging segment $ms(v)$ by intersecting two TRRs, trr_a with core $ms(a)$ and radius $|e_a|$, and trr_b with core $ms(b)$ and radius $|e_b|$, i.e., $ms(v) = trr_a \cap trr_b$. Figure 2 depicts an example of the construction of $ms(v)$. The following lemma can be used to show that if $ms(a)$ and $ms(b)$ are Manhattan arcs, then $ms(v)$ is also a Manhattan arc. Moreover, since for each sink s_i, we have that $ms(s_i)$ is a single point and thus a Manhattan arc, by induction all merging segments are Manhattan arcs.

Lemma 1 *: The intersection of two TRRs, A and B, is also a TRR and can be found in constant time. If $radius(A) + radius(B) = d(core(A), core(B))$, then $A \cap B$ is also a Manhattan arc.*

The proof of Lemma 1 is contained in [2].

Figure 3 illustrates a tree of merging segments. The leaves of the tree are all single points representing the sink locations $s_1, \ldots, s_8$, and the interior nodes of the tree are Manhattan arcs.

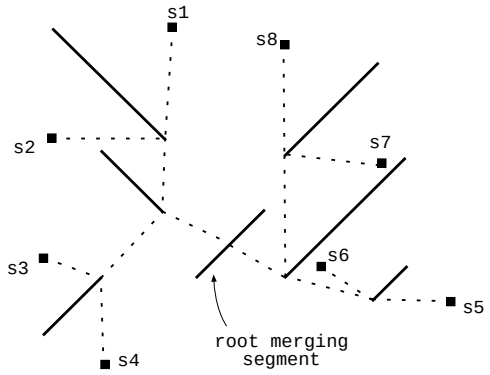

Figure 3: A tree of merging segments. Solid lines are merging segments; dotted lines indicate edges between merging segments.

Procedure Build_Tree_of_Segments
Input: Topology G; set of sink locations S
Output: Merging segments $ms(v)$ for each node v in G and edge lengths $|e_v|$ for each $v \neq s_0$
for each node v in G (bottom-up order)
  if v is a sink node,
    $ms(v) \leftarrow \{pl(v)\}$
  else
    Let a and b be the children of v
    Calculate_Edge_Lengths($|e_a|$,$|e_b|$)
    Create TRRs trr_a and trr_b as follows:
      $core(trr_a) \leftarrow ms(a)$
      $radius(trr_a) \leftarrow |e_a|$
      $core(trr_b) \leftarrow ms(b)$
      $radius(trr_b) \leftarrow |e_b|$
    $ms(v) \leftarrow trr_a \cap trr_b$
  endif

Figure 4: Constructing the tree of segments.

Figure 4 gives a precise description of the procedure Build_Tree_of_Segments, which constructs the tree of merging segments. Details of the Calculate_Edge_Lengths step depend on the delay model. For the linear model, the calculation is straightforward (see [2]). The calculation for the Elmore model can be found in [2][3][8]. Unless more wire is needed to balance delays between T_a and T_b, it must be that $|e_a| + |e_b| = d(ms(a), ms(b))$.

By Lemma 1, procedure Build_Tree_of_Segments requires constant time to compute each new merging segment, and linear time in the size of S to construct the entire tree of merging segments.

3.2 Phase II: Embedding of Nodes

Once the tree of segments has been constructed, the exact embeddings of internal nodes in the ZST are chosen in a top-down manner. For node v in topology G,

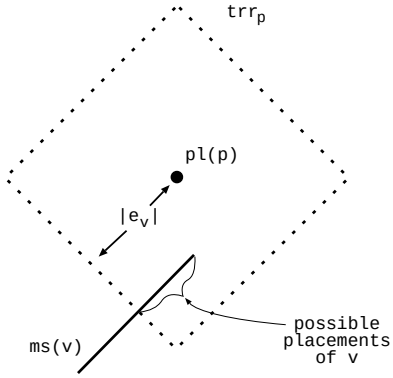

Figure 5: Finding the placement of v given the placement of its parent p.

Procedure Find_Exact_Placements
Input: Tree of segments TS containing $ms(v)$ and $|e_v|$ for each node v in G
Output: ZST $T(S)$
for each internal node v in G (top-down order)
  if v is the root
    Choose any $q \in ms(v)$
    $pl(v) \leftarrow q$
  else
    Let p be the parent node of v
    Construct trr_p as follows:
      $core(trr_p) \leftarrow \{pl(p)\}$
      $radius(trr_p) \leftarrow |e_v|$
    Choose any $q \in ms(v) \cap trr_p$
    $pl(v) \leftarrow q$
  endif

Figure 6: Creating the ZST by embedding internal nodes of the topology.

we select $pl(v)$ as follows: (i) if v is the root node, then any point in $ms(v)$ can be chosen as $pl(v)$;[3] and (ii) if v is an internal node other than the root, then v can be embedded at any point in $ms(v)$ that is at distance $|e_v|$ or less from $pl(p)$. (The merging segment $ms(p)$ was constructed such that $d(ms(v), ms(p)) \leq |e_v|$, so there must exist some choice of $pl(v)$ satisfying this condition.[4]) More specifically, the procedure creates a square TRR trr_p with radius e_v and with core equal to the placement of v's parent node p. The placement of v can be any point from $ms(v) \cap trr_p$ (see Figure 5). In Figure 3, the resulting placements for the tree of merging segments are indicated by the points where segments are connected by dotted lines. Figure 6 describes procedure Find_Exact_Placements, which performs the embedding of nodes from the tree of merging segments.

Since each instruction in Find_Exact_Placements is executed at most once for each node in G, and since the intersection of TRRs $ms(v)$ and trr_p can be found in constant time by Lemma 1, Find_Exact_Placements

[3] If the specification requires a fixed source location, s'_0, choose $pl(s_0) \in ms(s_0)$ with minimum distance from s'_0 and connect a wire directly from s'_0 to $pl(s_0)$.

[4] The distance can be less than $d(ms(v), ms(p))$ only when extra wire is used to merge v with its sibling w, i.e., when the merging cost for p is greater than $d(ms(v), ms(w))$.

requires time linear in the size of S. Hence, DME is a linear time algorithm overall.

4 Optimality of DME for Linear Delay

The DME algorithm is optimal in the linear delay regime (the proof of Theorem 1 is contained in [2]).

Theorem 1 *Given a set of sinks $S \subset \Re^2$ and a connection topology G, the DME algorithm produces a ZST T in the linear model with minimum cost over all ZSTs with topology G and sinks S.*

DME also produces the optimal ZST in the variation of the Zero Skew Clock Routing Problem where the position of the source is fixed. This extension to Theorem 1 is proved in [4].

Under the linear model, DME also minimizes the source-sink delay in a ZST. We now prove that given any input topology, DME will in fact construct a ZST with delay equal to one-half the diameter of the sink set S, which is the minimum feasible radius for any tree connecting S.

Define a *Manhattan disk* to be a TRR with a core consisting of a single point. In other words, a Manhattan disk is the set of all points within a prescribed radius of a central point. In the Manhattan plane, such a "disk" is actually shaped like a diamond (e.g., trr_p in Figure 5). Let $MD(s_i, r)$ denote the Manhattan disk with core $\{s_i\}$ and radius $r \geq 0$. The *diameter* of S is defined to be $min\{d(s_i, s_j) \mid s_i, s_j \in S\}$. Lemma 2 shows that it is feasible to construct a ZST for S with linear delay equal to one-half its diameter.

Lemma 2 *: Let d be the diameter of sink set S. Then*

$$\cap_{s_i \in S}[MD(s_i, d/2)] \neq \emptyset.$$

Proof: It is well known that the Manhattan metric after a 45 degree rotation is equivalent to the L_∞ metric, where $d[(x,y),(x',y')] = max\{|x-x'|, |y-y'|\}$. Hence we need only prove the lemma for the L_∞ metric, where TRRs are equivalent to rectangles with vertical and horizontal boundaries. Consider the smallest rectangle R with vertical and horizontal boundary lines that contains all points in S (after rotation). Let d be the diameter of S. Then both the width and height of R must be less than or equal to d (otherwise there would be two sinks s_i and s_j with $d(s_i, s_j) > d$). Consequently, the point at the center of R is within distance $d/2$ of all sinks in S, and is contained in $\bigcap_{s_i \in S}[MD(s_i, d/2)]$. □

The next lemma states that increasing the radius of two TRRs by a constant, δ, will increase the radius of their intersection by δ without changing its core.

Lemma 3 *: Let A and B be TRRs, and suppose $A \cap B = C \neq \emptyset$. Construct TRRs A' and B' such that for $\delta \geq 0$, $core(A') = core(A)$, $radius(A') = radius(A) + \delta$, $core(B') = core(B)$, and $radius(B') = radius(B) + \delta$. If $C' = A' \cap B'$, then $core(C') = core(C)$ and $radius(C') = radius(C) + \delta$.*

Proof: The lemma is obvious after transformation to the L_∞ metric, where TRRs become rectangles with vertical and horizontal boundaries. □

Theorem 2 *: For any sink set S and topology G, the DME algorithm will find a ZST with minimum feasible delay, equal to one-half the diameter of S.*

Proof: Let d equal the diameter of S. We assign a TRR, called $TRR(v)$, to each node $v \in G$ such that

- if v is a sink node, then $TRR(v) = MD(pl(v), d/2)$; and
- if v is an internal node with children a and b, then $TRR(v) = TRR(a) \cap TRR(b)$.

By Lemma 2, $TRR(s_0) = \cap_{s_i \in S}[MD(s_i, d/2)]$ is non-empty. Let s_j and s_k be two points in S such that $d(s_j, s_k) = d$. The intersection of $TRR(s_j) = MS(s_j, d/2)$ and $TRR(s_k) = MS(s_k, d/2)$ must have radius 0 (by Lemma 1), and so $TRR(s_0)$ must have radius 0.

For any node v, let $t_{LD}(v)$ be the linear delay (sum of edge lengths) from v to each of the sinks in the subtree of v constructed by the DME algorithm.

Fact: *For each node v in G, $core(TRR(v)) = ms(v)$ and $radius(TRR(v)) = d/2 - t_{LD}(v)$.*

We prove the Fact using induction on the maximum number of edges between v and sinks in its subtree. If v is a sink, then $core(TRR(v)) = \{v\} = ms(v)$, and

$$radius(TRR(v)) = d/2 = d/2 - t_{LD}(v).$$

If v is an internal node with children a and b, inductively assume that the Fact holds for a and b. In the linear delay model, we have that $t_{LD}(a) = t_{LD}(v) - |e_a|$. Hence,

$$\begin{aligned} radius(TRR(a)) &= d/2 - t_{LD}(a) \\ &= d/2 - t_{LD}(v) + |e_a| \end{aligned}$$

Similarly, $radius(TRR(b)) = d/2 - t_{LD}(v) + |e_b|$.

Consider the TRRs trr_a and trr_b constructed by procedure Build_Tree_of_Segments in Figure 4. By construction, $core(trr_a) = ms(a)$, $radius(trr_a) = |e_a|$, $core(trr_b) = ms(b)$, and $radius(trr_b) = |e_b|$. Thus,

$$radius(TRR(a)) = d/2 - t_{LD}(v) + radius(trr_a)$$

and

$$radius(TRR(b)) = d/2 - t_{LD}(v) + radius(trr_b)$$

In other words, $TRR(a)$ and $TRR(b)$ can be constructed from trr_a and trr_b, respectively, by adding the constant $d/2 - t_{LD}(v)$ to their radii. Consequently, Lemma 3 implies that $core(TRR(v)) = ms(v)$ and $radius(TRR(v)) = d/2 - t_{LD}(v)$. This proves the Fact.

Because $radius(TRR(s_0)) = 0$, we have that $t_{LD}(s_0) = d/2$, which proves the theorem. □

5 Suboptimality For Elmore Delay

While the experimental results in Section 6 clearly show the effectiveness of the DME algorithm in the Elmore delay model, examples exist for which DME does not give an optimal ZST under the Elmore model for a given topology [2][4]. The counterexample in [2][4] refutes the claim in [3] that the DME algorithm is optimal for any given routing topology under the Elmore model.

6 Results

We implemented the DME algorithm on Sun SPARC workstations in the C/UNIX environment. The code can be obtained from the authors. We used two sets of benchmarks: (i) the sink placements for the MCNC Primary1 and Primary2 benchmarks used in [5] and [6], and originally provided by the authors of [5]; and (ii) the sink placements for the five benchmarks r1 - r5 used in [8].

Our experimental results for linear delay are contained in Table 1. We applied the DME embedding algorithm to the topologies generated by the bottom-up, matching based method of Kahng, Cong and Robins (KCR) [6]. We compare our results with the original KCR results and with the Method of Means and Medians (MMM) of Jackson et al. [5]. The combined algorithm KCR+DME produced an average reduction in cost of 9% from the original KCR results and 16% from the MMM results. In the linear model, DME also produces trees with optimal source-sink *delay*. In our experiments, this optimal delay was on average 19% less than that of the KCR constructions.

	number of sinks	MMM cost	KCR cost	KCR +DME cost	reduction by KCR+DME from MMM (%)	reduction by KCR+DME from KCR (%)
P1	269	161.7	153.9	140.3	13.2	8.8
P2	603	406.3	376.7	350.4	13.8	7.0
r1	267	1,815	1,627	1,497	17.5	8.0
r2	598	3,625	3,349	3,013	16.9	10.0
r3	862	4,643	4,360	3,902	16.0	10.5
r4	1,903	9,376	8,580	7,782	17.0	9.3
r5	3,101	13,805	12,928	11,665	15.5	9.8
average					15.7	9.1

Table 1: Comparison of KCR+DME with other algorithms for the linear delay model, using MCNC benchmarks Primary1 (P1) and Primary2 (P2), and benchmarks r1 through r5 from Tsay.

	MMM cost	Tsay cost	Tsay +DME cost	KCR +DME cost	reduction by KCR+DME from MMM (%)	reduction by KCR+DME from Tsay (%)
P1	161.7			140.3	13.2	
P2	406.3			348.3	14.3	
r1	1,815	1,697	1,658	1,487	18.1	12.4
r2	3,625	3,432	3,368	3,020	16.7	12.0
r3	4,643	4,407	4,333	3,867	16.7	12.3
r4	9,376	8,866	8,694	7,713	17.7	13.0
r5	13,805	13,199	12,926	11,606	15.9	12.1
average					16.1	12.4

Table 2: Comparison of KCR+DME with other algorithms for the Elmore delay model. Results of Tsay's algorithm for benchmarks P1 and P2 were not available.

Similar improvements were obtained for Elmore delay on the same benchmarks, as shown in Table 2. The average reduction in wirelength was 16% versus the MMM results, and 12% versus the results of Tsay. Our results also indicate a very significant reduction in source-sink delay in the Elmore model: the combination of KCR+DME reduced Elmore delay by an average of 22% compared to the results of Tsay.

7 Conclusion

The Deferred-Merge Embedding (DME) algorithm offers many improvements over previous embedding schemes. DME constructs a highly flexible tree of merging segments which allows a choice among minimum-cost zero skew clock trees. Given any connection topology over the set of sink locations, DME always produces a tree with exact zero skew, and may thus be applied to previously generated clock trees in order to improve both wirelength and delay. Experiments show that applying DME to topologies generated by the algorithm of [6] results in wirelength reductions of 9% to 16% over [5] [6] [8]. Finally, under the linear delay model, DME yields *optimal* total wirelength for the topology and *optimal* source-sink delay overall.

8 Remarks and Acknowledgements

Most of the results in this paper also appear in [4], reflecting a collaboration between the present authors and the authors of [3] that arose after it was learned that the two groups had, through independent research, come up with essentially the same embedding approach. The authors are grateful to Dr. Ren-Song Tsay for providing benchmark data.

References

[1] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI* , Addison-Wesley, 1990.

[2] K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees With Minimum Wirelength," technical report UCLA CSD-920012, March 1992.

[3] T.-H. Chao, Y.-C. Hsu, and J.-M. Ho, "Zero Skew Clock Net Routing," to appear in *Proc. ACM/IEEE Design Automation Conf.*, 1992.

[4] T.-H. Chao, Y.-C. Hsu J.-M. Ho, K. D. Boese and A. B. Kahng, "Zero Skew Clock Routing With Minimum Wirelength," submitted to *IEEE Transactions on Computers and Systems*, 1992.

[5] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High Performance ICs," *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 573-579.

[6] A. B. Kahng, J. Cong, and G. Robins, "High-Performance Clock Routing Based on Recursive Geometric Matching," *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 322-327.

[7] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal Delay in RC Tree Networks," *IEEE Transactions on Computer-Aided Design* 2(3) July 1983, pp. 202-211.

[8] R. S. Tsay, "Exact Zero Skew," *IEEE Int. Conference on Computer-Aided Design*, 1991, pp. 336-339.