Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

ZERO-SKEW CLOCK ROUTING TREES WITH MINIMUM WIRELENGTH

K. Boese
A. Kahng

March 1992
CSD-920012

Zero-Skew Clock Routing Trees With Minimum Wirelength

*

Kenneth D. Boese and Andrew B. Kahng

Abstract

In the design of high performance VLSI systems, minimization of clock skew is an increasingly important objective. Clock routing trees should also have minimum wirelength in order to reduce system power requirements and deformation of the clock pulse at the synchronizing elements of the system. In this report, we present the Deferred-Merge Embedding (DME) algorithm, which embeds any given clock tree topology to minimize total wirelength while preserving zero clock skew. Extensive experimental results show that the algorithm yields zero-skew trees with 7% to 14% wirelength reduction over previous constructions in [12] [13] [18]. The DME algorithm may be applied with either the Elmore or linear delay model, and it yields *optimal* total wirelength for the linear delay regime.

1 Introduction and Preliminaries

Circuit speed is a primary consideration in the design of high performance VLSI designs. In a synchronous VLSI design, the circuit speed is increasingly limited by two factors: (i) delay on the longest path through combinational logic, and (ii) clock skew, i.e., the maximum difference in arrival times of the clocking signal among the synchronizing elements of the design. This is seen from the well known inequality governing the clock period of each clock signal net [2] [12]:

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

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 (assuming edge triggering), and t_{ds} is the propagation delay within the synchronizing elements. The term t_d can be further decomposed into $t_d = t_{d\_interconnect} + t_{d\_gates}$, where $t_{d\_interconnect}$ is the delay associated with the interconnect of the longest path through combinational logic, and t_{d_gates} is the delay through the combinational logic gates on this path. Increased switching speeds due to advances in VLSI fabrication technology imply that the terms t_{su}, t_{ds}, and t_{d_gates} all decrease significantly. Therefore, $t_{d\_interconnect}$ and t_{skew} become more dominant factors in determining circuit performance: Bakoglu [2] has noted that t_{skew} may account for over 10% of the system cycle time in high-performance systems. With this in mind, a number of researchers have recently studied the clock skew minimization problem.

*This work was supported in part by NSF MIP-9110696, ARO DAAK-70-92-K-0001, and a GTE Graduate Fellowship.
Author affiliations: Dept. of Computer Science, UCLA, Los Angeles, CA 90024-1596.

For building block design styles, Ramananathan and Shin [14] proposed a clock distribution scheme which applies when the blocks are hierarchically organized. The number of blocks at each level of the hierarchy is assumed to be small since the algorithm exhaustively enumerates all possible clock routings and clock buffer optimizations. Burkis [5] and Boon et al. [4] also proposed hierarchical clock tree synthesis approaches involving geometric clustering and buffer optimization at each level. General clock tree resynthesis or reassignment methods were used by Fishburn [9] and Edahiro [7] to minimize the clock period while avoiding hazards or race conditions; [9] employed a mathematical programming formulation, while [7] employed computational geometric techniques within a clustering-based heuristic. All of these methods are essentially limited to small problem sizes, either by algorithmic complexity or by reliance on a strong hierarchical clustering. By contrast, we are interested in clock tree synthesis for "flat" designs with many sinks (synchronizing elements); such instances arise for large standard-cell and sea-of-gates designs, as well as multi-chip module packages.

Clock tree construction for designs with many clock sinks was first attacked by the H-tree method, which was used in regular systolic arrays by Bakoglu and other authors [1] [6] [10] [19]. While the H-tree structure can significantly reduce clock skew [6] [19], it is applicable only when all of the synchronizing components are identical in size and are placed in a symmetric array. The first robust clock tree construction for cell-based layouts is due to Jackson, Srinivasan and Kuh [12]: their "method of means and medians" (MMM) algorithm generates a topology by recursively partitioning the set of sinks into two equal-size subsets, then connecting the center of mass of the entire set to the centers of mass of the two subsets. Although it was shown in [12] that the maximum difference in pathlength from the root to different synchronizing components is bounded by $O(\frac{1}{\sqrt{n}})$ on average, [13] showed small examples for which the source-sink pathlengths in the MMM solution may vary by as much as half the chip diameter. In some sense, this reflects an inherent weakness in the top-down approach, which can commit to an unfortunate topology early on in the construction. Kahng, Cong and Robins [13] gave a bottom-up matching approach to clock tree construction: their method eliminates all skew in terms of source-sink pathlengths, while using 5%-7% less total wirelength than the MMM algorithm. However, as [13] focused primarily on pathlength balancing, their method addresses clock skew minimization only in the sense of the *linear* delay model. Tsay [18] subsequently gave a similar bottom-up algorithm that achieved zero skew trees with respect to the Elmore delay model [8] [15]. In the same spirit as the method of [13], Tsay's method combines pairs of zero skew trees at "tapping points" (the roots of the recursively merged subtrees, analogous to the "balance points" in [13]) to yield larger zero skew trees.

Both the top-down method of [12] and the bottom-up methods of [13] [18] concentrate on the problem of computing a good heuristic clock tree *topology*. However, this work essentially disregards total wirelength considerations (e.g., [18] adds redundant wiring as needed to match delays). In other words, they do not effectively address the associated problem of finding a minimum-cost *embedding* of the heuristic topology. Indeed, these previous methods are highly inflexible in that they permanently embed each internal node of the tree as soon as it becomes defined. Disregarding wirelength may not be practical, since excess interconnect will not only increase layout area but also result in greater tree capacitance, thus requiring more power for distribution of the clock signal.

In this report, we propose a new approach which achieves exact zero clock skew while significantly

reducing the total wirelength in the clock tree, i.e., for any given topology, we can achieve an effective embedding. The basic idea of our algorithm is to *defer* the embedding of internal nodes in the 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, our Deferred-Merge Embedding (DME) algorithm will begin with an initial clock tree computed by any previous method, then maintain zero clock skew while reducing the wirelength. In regimes where the linear delay model applies, our method produces the optimal (minimum-cost) zero skew clock tree with respect to the prescribed topology, and this tree will also enjoy the minimum possible source-terminal delay.

Experimental results in Section 4 below show that the DME approach is highly effective with the linear delay model. We achieve average savings in total clock tree cost of 13% over the MMM algorithm [12] and 7% to 9% over the method of [13]. Our methods are quite robust, and extend to "prescribed skew" formulations as well as more general optimizations of topologies for both clock routing and global routing.

The remainder of this report is organized as follows. In Section 2, we formalize the minimum-cost zero skew clock routing problem and also establish the linear and Elmore delay models that are used in the subsequent discussion. Section 3 presents our main results. These include: (i) the Deferred-Merge Embedding (DME) algorithm for efficiently embedding a given topology and (ii) application of the DME algorithm to both the linear and Elmore delay regimes; Section 4 gives experimental results and comparisons with previous work. Results contained in this report have been submitted to the IEEE International Conference on ASIC (1992). The summary submitted to that conference for review is included in Appendix A.

2 Problem Formulation

The placement phase of physical layout determines positions for the synchronizing elements of the circuit design, which we call the *sinks* of the clock net. A finite set of sinks, denoted by $S = \{s_1, s_2, \ldots, s_n\} \in \Re^2$, specifies an instance of the clock routing problem. We say that a *connection topology* is an unweighted 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] 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 therefore 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 set of all children of v is denoted by $children(v)$. The wirelength, or cost, of the edge e_v is denoted by $|e_v|$. The cost of T is the total wirelength of the edges in T.

Given a clock tree $T(S)$, we use $t_d(s_0, s_i)$ to denote the signal propagation time, or *delay*, on the the unique path from the source s_0 to the sink s_i; the collection of edges in this path is denoted by $path(s_0, s_i)$.[2] The *skew* of $T(S)$ is the maximum value of $|t_d(s_0, s_i) - t_d(s_0, s_j)|$ over all sink pairs $s_i, s_j \in S$, and $T(S)$ is a *zero skew clock tree* (ZST) if its skew is zero. The zero skew clock routing problem is thus stated as

[1] Note that the binary tree representation suffices to capture arbitrary Steiner routing topologies.

[2] Note that a path from node v to node w, where v and w are on some source-sink path, will not contain e_v.

follows:

Given a set S of sinks, construct a zero skew clock tree $T(S)$ with minimum cost.

Our work is concerned with the variant of the zero skew clock routing problem where a connection topology G is prescribed, and we wish to find the minimum cost zero skew clock tree that has topology G.

Observe that the notion of a zero-skew clock tree is well-defined only when we can evaluate 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.[3] Using equations such as those of Rubinstein et al. [15], one may achieve tight upper and lower bounds on delay in a distributed RC tree model of the clock routing. However, two simpler approximations of RC delay are more easily computed and optimized during the clock tree construction. These are the *linear delay* model and the *Elmore delay* model, which are appropriate to clock tree synthesis in different technological regimes.

2.1 Delay Models

2.1.1 Linear Delay

In the linear delay model, the delay along $path(s_0, s_i)$ is proportional to the length of the path and is independent of the remainder of the connection topology. We may express the path delay $t_{LD}(s_0, s_i)$ as

$$t_{LD}(s_0, s_i) = \sum_{e_v \in path(s_0, s_i)} |e_v|.$$

More generally, the linear delay between any two nodes u and w in a source-sink path is

$$t_{LD}(u, w) = \sum_{e_v \in path(u, w)} |e_v|.$$

The linear delay model is particularly appropriate for emerging substrate interconnect technologies on multichip module (MCM) packages [17], e.g., low resistivity thin-film interconnect. The linear delay model also has added relevance for MCMs because the MCM approach achieves its performance wins by integrating multiple die on a single package, i.e., interchip delay is significantly reduced. For yield and ease of system design, individual ICs in an MCM design are often relatively slow compared with leading-edge monolithic packages, and the linear delay model remains a reasonable approximation. Below, we show that for the linear delay model, our DME algorithm gives the optimal solution to the zero skew clock routing problem for *any* prescribed connection topology. However, with higher ASIC system speeds the Elmore delay model gives a more accurate delay approximation.

[3] The global routing phase of layout will typically consider the clock and power/ground nets for preferential assignment to (dedicated) routing layers. We assume that the interconnect delay parameters are the same on all metal routing layers, and we ignore via resistances. Thus, wirelength becomes a valid measure of the RC parameters of interconnections.

2.1.2 Elmore Delay

Let α and β respectively denote the resistance and capacitance per unit length of interconnect, so that the resistance r_{e_v} and capacitance c_{e_v} of e_v are respectively given by $\alpha \cdot |e_v|$ and $\beta \cdot |e_v|$. For each sink s_i in the tree $T(S)$, there is a loading capacitance c_{L_i} which is the input capacitance of the functional unit driven by s_i.

We let T_v denote the subtree of $T(S)$ that has v as its root, and we let C_v denote the tree capacitance of T_v; this is equal to the total node capacitance of T_v. C_v can be calculated from the node capacitance of v and the summation of the capacitances of its children, yielding the following recursive formula:

$$C_v = \begin{cases} c_{L_i} & \text{if } v \text{ is a sink node } s_i \\ \sum_{w \in children(v)} c_{e_w} + C_w & \text{if } v \text{ is an internal node} \end{cases}$$

According to [16] [15] [8], the Elmore delay $t_{ED}(s_0, s_i)$ can be calculated by the following formula (see [18] for a discussion of related circuit models):

$$t_{ED}(s_0, s_i) = \sum_{e_v \in path(s_0, s_i)} r_{e_v} C_v.$$

More generally, the delay time between any two vertices u and w on a source-sink path is given by

$$t_{ED}(u, w) = \sum_{e_v \in path(u, w)} r_{e_v} C_v.$$

The Elmore delay is additive: if v is a vertex on the u-w path, then $t_{ED}(u, w) = t_{ED}(u, v) + t_{ED}(v, w)$. For example, if v is a child of u and w is a sink s_i, then $t_{ED}(u, s_i) = r_{e_v} C_v + t_{ED}(v, s_i)$. A sink node s_i may be treated as a ZST with capacitance c_{L_i} and delay zero.

3 Main Results

At a high level, we divide the construction of the ZST into into (i) generation of a connection topology, and (ii) embedding of the connection topology in the Manhattan plane. This report presents the Deferred-Merge Embedding (DME) algorithm, which computes a wire-efficient embedding of a given topology. The approach is successful with both the linear and Elmore delay models; in the linear delay model it will produce the minimum cost ZST for the given topology.

3.1 The Deferred-Merge Embedding (DME) Algorithm

The Deferred-Merge Embedding algorithm embeds interior nodes of the connection topology G via a two-phase process. First, we process the topology in bottom-up order, and construct a tree of line segments which represents possible placements of interior nodes in the ZST T but defers the actual node embeddings. Then, a top-down phase resolves the exact embeddings of the nodes of T.

3.1.1 Bottom-Up Phase: The Tree of Segments

Given the set of sink locations S and the connection topology G, we construct a *tree of merging segments.* Each merging segment is associated with a node of G and represents a set of possible placements of that node. The tree of segments is constructed in a bottom-up order, because the definition of each segment requires previous definition of its two children.

Let v be a node in connection topology G with children a and b. Suppose T_a and T_b are zero-skew embeddings of the subtrees rooted at a and b, respectively. Then a *merging point* p of v is a placement of v which allows T_a and T_b to be merged into a zero-skew tree T_v with minimum additional wire. Let e_a and e_b be the edges connecting T_a and T_b to p. The edge lengths $|e_a|$ and $|e_b|$ are called the *merging distances* between v and a and between v and b, respectively. The calculation of the merging distances depends on the delay model used and will be described in Sections 3.2.1 and 3.3.1 below.

The additional wirelength used to merge T_a and T_b equals $|e_a| + |e_b|$ and is called their *merging cost.* If the merging cost is minimized, either the merging cost will equal the distance $d(a, b)$, or, if additional wire is needed to balance the delays, one merging distance will equal 0 and the other will be greater than $d(a, b)$. A *merging segment* between points a and b is the set of all merging points between T_a and T_b.

Let $d(p, q)$ denote the Manhattan distance between two points p and q. Define the distance between two sets of points P and Q, $d(P, Q)$ to be

$$min\{d(p, q) \mid p \in P \text{ and } q \in Q\}$$

We define a *Manhattan arc* as a line segment with slope +1 or -1. The collection of points within a fixed distance of a Manhattan arc is rectangular region whose boundary lines are also Manhattan arcs. Such a region is called a *tilted rectangular region* or *TRR.* (See Figure 1 for an example of a TRR.) The Manhattan arc at the center of a TRR is called its *core*, and is formally defined as the subset of the TRR that is of maximum distance from its boundary. The *radius* of a TRR is the distance between its core and its boundary.

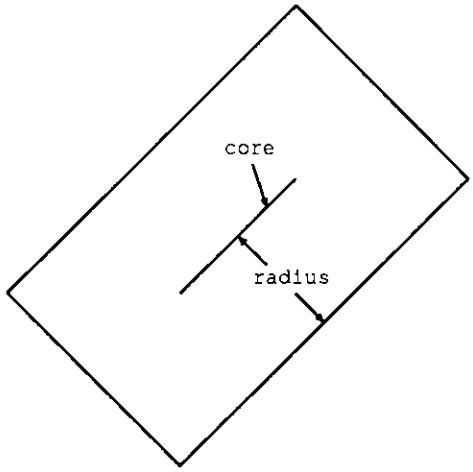

Figure 1: An example of a TRR with core and radius as indicated.

The roots of subtrees T_a and T_b can be anywhere on their respective merging segments. Call these

segments $ms(a)$ and $ms(b)$. Then every point p on the merging segment $ms(v)$ between T_a and T_b, $ms(v)$, must be within distance $|e_a|$ of $ms(a)$ and within distance $|e_b|$ of $ms(b)$. Consequently, if we build TRRs trr_a with core $ms(a)$ and radius $|e_a|$ and trr_b with core $ms(b)$ and radius $|e_b|$, then $ms(v) \subset trr_a \cap trr_b$. Moreover, because any point p with $d(p, ms(a)) \leq |e_a|$ and $d(p, ms(b)) \leq |e_b|$ will be a merging point between T_a and T_b with minimum merging cost, $ms = trr_a \cap trr_b$.

Figure 2 contains a more precise description of the algorithm to construct the tree of merging segments. In the figure, Figures 3 and 4 illustrate the algorithm for the two cases where (a) the merging cost equal $d(ms(a), ms(b))$ and (b) the merging cost is greater than $d(ms(a), ms(b))$.

Procedure Build_Tree_of_Segments
Input: Topology G
Output: Merging Segments $ms(v)$ for each node v in G
for each node v in G (bottom-up order)
  bf if v is a terminal node,
    $ms(v) \leftarrow \{placement(v)\}$
  else
    Let a and b be the children of v.
    Calculate_Merging_Distances($|e_a|$,$|e_b|$);
    Create TRR's 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 2: Algorithm to construct tree of segments.

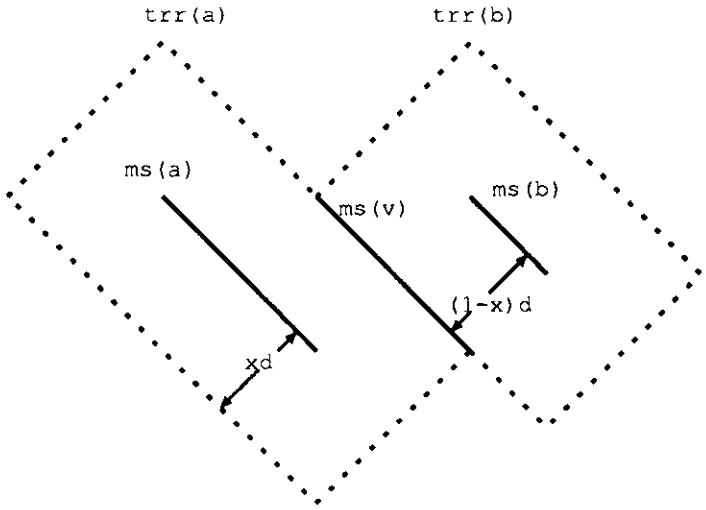

Figure 3: Procedure Build_Tree_of_Segments: construction of merging segment $ms(v)$ if the merging cost equals $d(ms(a), ms(b))$.

The next lemma proves that merging segments will always be Manhattan arcs (although they may sometimes be single points). The lemma also states that $trr_a \cap trr_b$ can computed in constant time. Con-

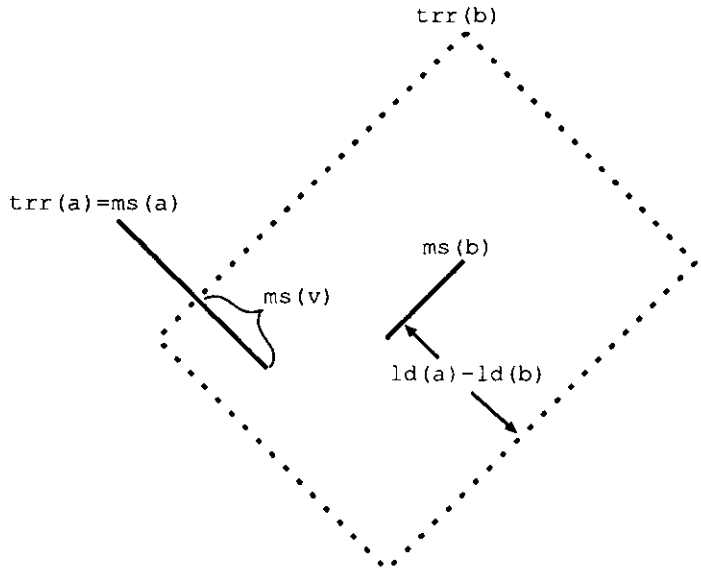

Figure 4: Procedure Build_Tree_of_Segments: construction of merging segment $ms(v)$ if merging cost is greater than $d(ms(a), ms(b))$.

sequently, the procedure Build_Tree_of_Segments requires constant time to compute each merging segment, and linear time to construct the entire tree of segments.

Lemma 1 *The intersection of two TRRs, R_1 and R_2, is also a TRR and can be found in constant time. If $radius(R_1) + radius(R_2) = d(core(R_1), core(R_2))$, then $R_1 \cap R_2$ is a Manhattan arc.*

Proof: The lemma follows readily if we rotate the plane by 45 degrees so that the boundaries of the TRRs are vertical and horizontal line segments. (See Figure 5.) Let R_1 and R_2 be the two TRRs after rotation, and let their boundary lines be as follows:

- R_1: ($a_1 \leq a_2$ and $b_1 \leq b_2$)

$$\begin{aligned} x &= a_1 \\ x &= a_2 \\ y &= b_1 \\ y &= b_2 \end{aligned}$$

- R_2: ($a_3 \leq a_4$ and $b_3 \leq b_4$)

$$\begin{aligned} x &= a_3 \\ x &= a_4 \\ y &= b_3 \\ y &= b_4 \end{aligned}$$

Then $R_1 \cap R_2$ is a rectangular region with boundary lines

$$\begin{aligned} x &= max(a_1, a_3) \\ x &= min(a_2, a_4) \\ y &= max(b_1, b_3) \\ y &= min(b_2, b_4) \end{aligned}$$

The construction above of $R_1 \cap R_2$ requires a constant number of steps. Since rotating a rectangle by 45 degrees also requires only constant time, the intersection of two TRRs can be found in constant time.

If $radius(R_1) + radius(R_2) = d(core(R_1), core(R_2))$, then decreasing the radius of either R_1 or R_2 must cause their intersection to be empty. (Otherwise, we could form a path between $core(R_1)$ and $core(R_2)$ with length less than $d(core(R_1), core(R_2))$.) Consequently, $R_1 \cap R_2$ must have zero width and be a segment or a single point. Since it is also a TRR, it must be a Manhattan arc. □

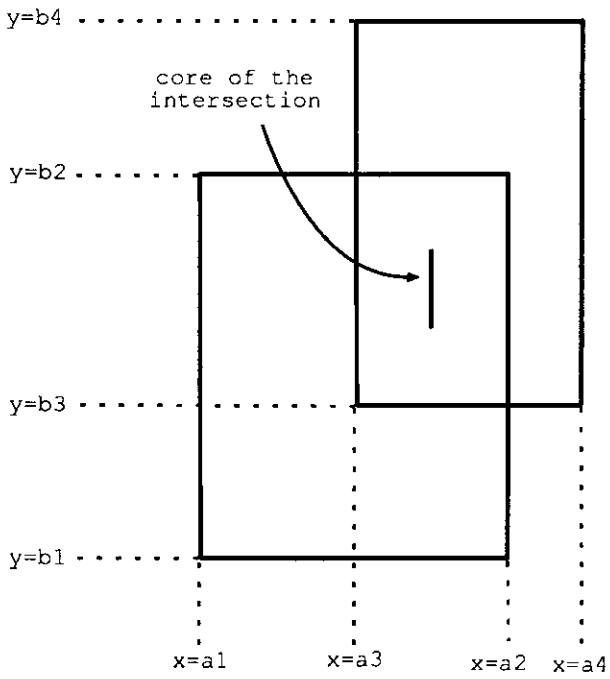

Figure 5: Intersection of TRR's after 45 degree rotation. In this example, the intersection is defined by boundary lines $x = a_3$, $x = a_2$, $y = b_3$, and $y = b_2$.

3.1.2 Top-Down Embedding of Nodes

Once the tree of segments has been constructed, the exact placements of interior nodes in the ZST T are chosen in a top-down manner. For node v in topology G, we select the embedding of v, $placement(v)$, as follows. If v is the root node, then any point in $ms(v)$ can be chosen as $placement(v)$. If v is an interior node other than the root, then v can be embedded at any point in $ms(v)$ that is within distance $|e_v|$ of the

placement of v's parent p. Merging segment $ms(p)$ is constructed so that $d(ms(v), ms(p)) \leq |e_v|$, which guarantees that there exists a point $placement(v)$ within distance $|e_v|$ of $placement(p)$.

Figure 6 contains the algorithm to embed interior nodes given the tree of segments. The algorithm creates a square TRR trr_p with radius equal to the length of e_{v_i} and with core equal to the placement of v_i's parent node v_p. The placement of v_i can be any point from $ms(v_i) \cap trr_p$. (See Figure 7).

```
Procedure Find_Exact_Placements
Input: tree of segments TS containing ms(v) for each v in G
Output: ZST T
for each interior node v in G (top-down order)
   if v is the root
      choose any q ∈ ms(v)
      placement(v) ← q
   else
      Let p be the parent node of v.
      construct trr_v and trr_p as follows
         trr_v ← ms(v); /* radius(trr_v) = 0 */
         core(trr_p) ← {p}
         radius(trr_p) ← edge_length(v)
      choose any q ∈ trr_v ∩ trr_p
      placement(v) ← q
   endif
```

Figure 6: Pseudo-code algorithm for embedding interior nodes in the ZST

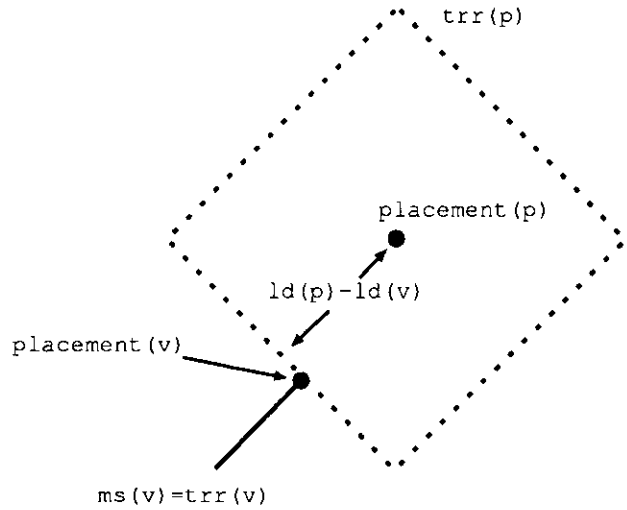

Figure 7: Procedure Find_Exact_Placements: finding the placement of v from the placemnt of its parent p.

3.2 Application of DME to Linear Delay

3.2.1 Calculating Edge Lengths

Calculating edge lengths is very straightforward in the linear delay model. Let a and b be two siblings in topology G with merging segments $ms(a)$ and $ms(b)$. Suppose $t_{LD}(a)$ and $t_{LD}(b)$ are the delays between a and b and each of the leaf nodes in their respective subtrees. Let v be the parent of a and b.

Zero skew at v requires that

$$t_{LD}(a) + |e_a| = t_{LD}(b) + |e_b|$$

If $|t_{LD}(a) - t_{LD}(b)| \leq d(ms(a), ms(b))$, then the merging cost is minimized with $|e_a| + |e_b| = d(ms(a), ms(b))$. Hence,

$$|e_a| = \frac{d(ms(a), ms(b)) + t_{LD}(b) - t_{LD}(a)}{2}$$

and

$$|e_b| = \frac{d(ms(a), ms(b)) + t_{LD}(a) - t_{LD}(b)}{2}$$

If on the other hand, $|t_{LD}(a) - t_{LD}(b)| > d(ms(a), ms(b))$, then the merging cost will be minimized with one of the edge lengths set to 0. If $t_{LD}(a) > t_{LD}(b)$, then $|e_a| = 0$ and $|e_b| = t_{LD}(a) - t_{LD}(b)$. Similarly, if $t_{LD}(a) < t_{LD}(b)$, then $|e_a| = t_{LD}(b) - t_{LD}(a)$ and $|e_b| = 0$.

3.2.2 Optimality of DME for Linear Delay

We have proved the following theorem asserting the optimality of DME for the linear delay model.

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

The proof of Theorem 1 requires the following three lemmas. First, define a ***straight-line path*** between two points x and y as any path of shortest length between them (in the Manhattan metric). If x and y are not on the same horizontal or vertical line, then there will be an infinite number of straight line paths between x and y. Define the *projection area* $PA(x, Q)$ from a point x through a set of points Q as the set of all points p for which there exists a straight line path from x to p that passes through Q. (Q must be between p and x.) Figure 8 contains an example of the projection area from a point x through a Manhattan arc Q.

Lemma 2 *Let ms be a merging segment between the two points p and q. Then*

$$PA(p, ms) \cup PA(q, ms) = \Re^2.$$

In other words, the union of the two projection areas from p and q, respectively, through their merging segment, ms, is the entire plane.

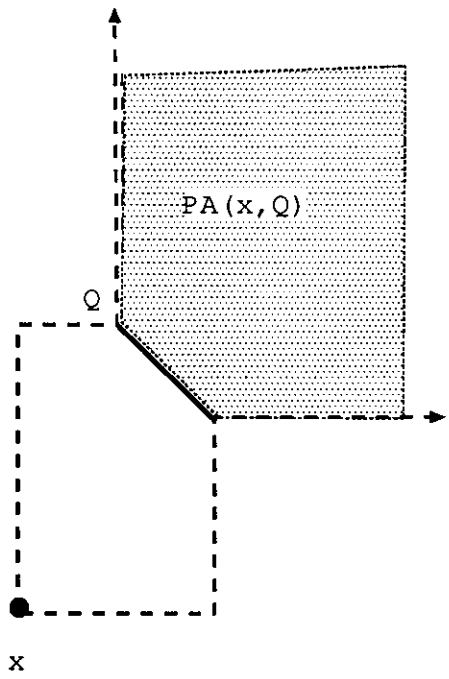

Figure 8: Projection areas under the Manhattan and Euclidean metrics.

Proof: Merging segment ms is constructed as the intersection of two TRRs, trr_p and trr_q, such that $core(trr_p) = \{p\}$, $core(trr_q) = \{q\}$, and

$$\begin{aligned} radius(trr_p) &= x * d(p,q) \\ radius(trr_q) &= (1-x) * d(p,q) \end{aligned}$$

where $0 \leq x \leq 1$. If $x = 1$ or $x = 0$, the lemma is immediately true, since $PA(p,\{p\}) = PA(q,\{q\} = \Re^2$. Let e_1 and e_2 be the two endpoints of merging segment ms. If $0 < x < 1$ then we need to consider the two cases depicted in Figure 9:

(a) e_1 and e_2 are both corners of the same TRR, either trr_p or trr_q.

(b) e_1 and e_2 are corners of different TRR's: e_1 a corner of trr_a and e_2 a corner of trr_b.

Define a *ray* $p_1\vec{p}_2$ from point p_1 through point p_2 as the half-line with endpoint p_1 that extends through p_2. In case (a), we can assume without loss of generality that e_1 and e_2 are both corners of trr_p. Consequently, the staight-line paths between p and e_1 and e_2 must be a vertical and a horizontal line. In Figure 9(a) it is evident that $PA(p,e_1)$ is the half plane with border line e_1p_1 and $PA(p,e_2)$ is the half plane bordered by line e_2p_2. Moreover, $PA(p,ms)$ is the infinite region separated from p by ray $e_1\vec{p}_1$, segment ms, and ray $e_2\vec{p}_2$. $PA(q,ms)$ is the region separated from q by the same border. Consequently, $PA(p,ms) \cup PA(q,ms)$ is the entire plane.

Similarly, in case (b) (Figure 9(b)), $PA(p,ms)$ is the infinite region separated from p by $e_1\vec{p}_1$, ms, and $e_2\vec{p}_2$. $PA(q,ms)$ is the region separated from q by the same border. So again, $PA(p,ms) \cup PA(q,ms) = \Re^2$. □.

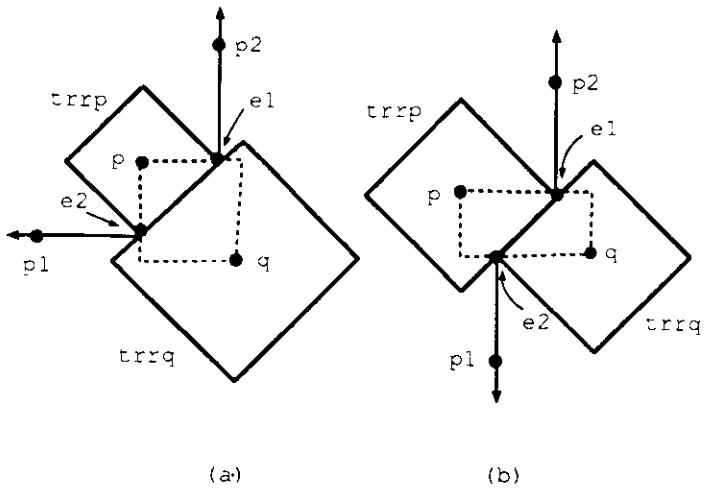

Figure 9: Two cases to consider in the proof of Lemma 4.

Lemma 2 is used in the proofs of Lemmas 3 and 4. Lemma 3 shows that in the optimal ZST the placement of a node must be in its merging segment. Lemma 4 demonstrates that the placement of two sibling nodes must be a nearest pair of points between their merging segments. Let $placement(T, v)$ be the embedding of node $v \in G$ for ZST T.

Lemma 3 *Given v an interior node in a ZST $T(S)$ with topology G. Suppose that a and b are the children of v, and the subtrees of T rooted at a and b can be generated using DME for some placement of v on $ms(v)$. Then if $q \equiv placement(T, v) \notin ms(v)$, then a new ZST T' can be constructed such that $cost(T') < cost(T)$, $placement(T', v) \in ms(v)$, and the delay at point q remains unchanged.*

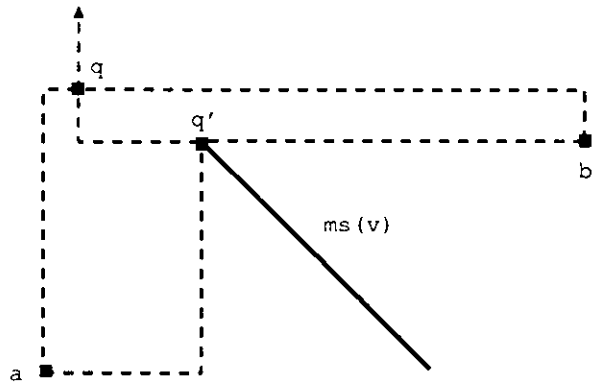

Figure 10: Optimal placement of v must be on $ms(v)$.

Proof: Consider Figure 10. Let a and b be the placements in T of v's children. By the Lemma 2, there exists a point q' on $ms(v)$ such that there is a straight-line path from a to q or from b to q. that passes through q'. Without loss of generality, assume that this path is from b to q. Now, consider the ZST with source q that is constructed by cutting T at q. Because $b - q' - q$ is a straight-line path, the segment bq in T can be replaced with segments bq' and qq' without increasing the delay between b and q. So,

$$t_{LD}(a) + length(T, aq) \quad = \quad t_{LD}(b) + length(T, bv)$$

$$\geq \quad t_{LD}(b) + d(b, q') + d(q', q)$$

where $length(T, aq)$ is the edge length between points a and q in T. Furthermore, since $ms(v)$ was chosen to balance the delay at points a and b, we must have that

$$t_{LD}(a) = t_{LD}(b) + d(b, q') - d(a, q')$$

and, thus

$$length(T, aq) \geq d(a, q') + d(q', q).$$

Hence, moving the placement of v from q to q' can reduce the cost of the tree by at least $d(q', q)$, while preserving the leaf distance at point q. □

The next lemma shows that in an optimal ZST, two sibling nodes must not only be chosen from their respective merging segments, but must also satisfy the distance constraints contained in procedure Find_Exact_Placements.

Lemma 4 *Suppose that in ZST T, a and b are two sibling nodes with parent v. Suppose also that the subtrees of T rooted at a and b can be generated using the DME algorithm. If $d(a, b) > d(ms(a), ms(b)$ and $d(a, b) > |t_{LD}(a) - t_{LD}(b)|)$, then a new tree T' can be constructed $cost(T') < cost(T)$ with unchanged delay at placement(T, v).*

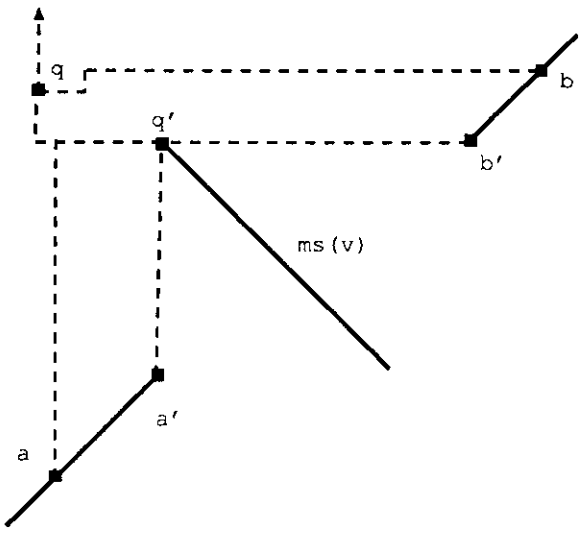

Figure 11: Optimal placement of siblings must be close enough to satisfy procedure Find_Exact_Placements.

Proof: (See Figure 11). Let a' and b' be placements of a and b on $ms(a)$ and $ms(b)$ that are within the correct distance:

$$d(a', b') \leq MAX(d(ms(a), ms(b)), |t_{LD}(a) - t_{LD}(b)|),$$

Let $q = placement(T, v)$. Then by Lemma 2, either $q \in PA(b', ms(v))$ or $q \in PA(a', ms(v))$. Assume without loss of generality that $q \in PA(b', ms(v))$. Then there is a straight line path between b' and q that passes through $ms(v)$ at some point q'. Furthermore, $d(b', ms(v) \leq d(b, ms(v))$, and consequently

$$d(b', q) \leq d(b, q).$$

Because the choice of a root node in DME does not change the cost of the resulting tree, we can move a to a' and b to b' without affecting the cost of their subtrees. Consequently

$$d(b', q') + d(q', q) = d(b', q) \geq d(b, q).$$

This implies that tree T can be modified by moving node v from q to q' while leaving the delay at point q unchanged. Moreover, the cost of the new tree T' is less than that of T by at least $d(q'q)$ □

Proof of Theorem 1: The proof is by contradiction. Suppose ZST T has minimum cost for point set S and topology G but cannot be produced using the DME algorithm. Then there must a non-empty set of nodes N such that $v \in N$ if and only if

- $placement(T, v) \notin ms(v)$; or
- if a is the sibling of v,

$$d(placement(T, v), placement(T, a)) > MAX(d(ms(a), ms(b)), |t_{LD}(v) - t_{LD}(b)|),$$

Let v' be a node in N with greatest depth in G. Let a' be the sibling of v'. Then since v' has maximum depth, all of the descendents of v' and a' can be produced using DME. Consequently, by Lemmas 3 and 4, v' can also be produced by the embedding algorithm, which contradicts the definition of set N. □

3.2.3 Experimental Results for Linear Delay

As a test of the tree cost performance of DME for linear delay, we ran the algorithm on placements of the MCNC benchmarks Primary1 and Primary2 used in [12] and [13]; these were originally provided by the authors of [12]. Primary1 contains 269 sink nodes and Primary2 contains 603 sink nodes. We used the topologies produced by Kahng, Cong, and Robins (KCR) [13] using a bottom-up, matching based method. We compared our results with the Method of Means and Medians (MMM) of Jackson, et al [12] and with KCR [13]. The results are presented in Table 1. Using the same topologies, the DME algorithm reduced the cost of trees produced by the KCR algorithm from 7% to approximately 9%. The overall reduction from MMM was between 13% and 14%. The embedding algorithm also produced a significant improvement in the delay time between source and sink nodes. For Primary1 the delay time was reduced from 6.6 in KCR to 5.2. For Primary2 the reduction was from 10.8 to 9.8.

	MMM cost	KCR cost	KCR + DME cost	cost reduction from MMM (%)	cost reduction from KCR (%)
Primary1	161.7	153.9	140.3	13.2	8.8
Primary2	406.3	376.7	350.4	13.8	7.0

Table 1: Comparison of the embedding algorithm in the linear delay model on industry benchmarks Primary 1 and Primary 2. We used the topology produced by KCR. Note that the MMM algorithm produced trees with standard deviation in pathlength of 0.29 for Primary1 and 0.74 for Primary2. KCR and KCR+EMB constructed trees with zero skew.

3.3 Application to Elmore Delay

3.3.1 Calculating Edge Lengths in the Elmore Delay Model

In this section, we present the calculation of edge lengths needed to merge two ZST's with minimum merging cost. These calculations are due to Tsay [18]. Let T_a and T_b be two ZST's with capacitance $C1$ and $C2$, respectively, and delay $t_1 = t_{ED}(a)$ and $t_2 = t_{ED}(b)$. Suppose v is a merging point with minimum merging cost. Let T_v be the ZST with root v and edges e_a and e_b connecting T_a and T_b.

From the definition of Elmore delay, we have that $t_{ED}(v,a) = r_{e_a}(\frac{1}{2}c_{e_a} + C_1) + t_1$. So merging point v satisfies the following equation:

$$r_{e_a}(\frac{1}{2}c_{e_a} + C_1) + t_1 = r_{e_b}(\frac{1}{2}c_{e_b} + C_2) + t_2. \tag{1}$$

Let $d(a,b) = \kappa$. Suppose that T_a and T_b can be merged with merging cost κ; in other words, $|e_a| = x$ and $|e_b| = \kappa - x$ for $0 \leq x \leq \kappa$. Then the resistances r_{e_a} and r_{e_b} are αx and $\alpha(\kappa - x)$, respectively. Also, the capacitances c_{e_a} and c_{e_b} are βx and $\beta(\kappa - x)$, respectively. By substituting these equalities into Equation 1 and solving for x, we have that

$$x = \frac{t_2 - t_1 + \alpha\kappa(C_2 + \frac{1}{2}\beta\kappa)}{\alpha(C_1 + C_2 + \beta\kappa)} \tag{2}$$

case 1: If $0 \leq x \leq \kappa$, then there exists a zero skew merging point of T_a and T_b located within the boundaries of a and b. The merging distance is solved by Equation 2 and the merging cost is κ.

case 2: If $x < 0$ or $x > \kappa$, then the assumption of merging cost κ results in a negative edge length for either e_a or e_b. In this case, an extended distance κ' is required to balance the difference in the delays of the two trees. If $x < 0$, which means $t_1 > t_2$, we choose a as the merging point and set $|e_a| = 0$ and $|e_b| = \kappa'$. By Eq. 1, we have that

$$t_1 = \alpha\kappa'(\frac{1}{2}\beta\kappa' + C_2) + t_2$$

and we use the quadratic formula to solve for κ':

$$\kappa' = \frac{((\alpha C_2)^2 + \alpha\beta(t_1 - t_2))^{\frac{1}{2}} - \alpha C_2}{\alpha\beta}$$

Similarly, if $x > \kappa$, we choose node b as the merging point and

$$\kappa' = \frac{((\alpha C_1)^2 + \alpha\beta(t_2 - t_1))^{\frac{1}{2}} - \alpha C_1}{\alpha\beta}$$

In this case, the merging cost is κ'.

The above analysis shows that a zero skew merging point between two ZSTs can always be found.

3.3.2 Counter-Example for Elmore Delay

For the linear delay model, DME produces a minimum cost ZST for a given topology. The deferred placement of interior nodes in our algorithm insures that our placement algorithm will reduce the cost over placement methods that merge subtrees with minimum additional cost without deferring the exact placements.

However, for some sink sets and topologies, our embedding scheme will not be optimal in the Elmore delay model. This is demonstrated by the ZST's T and T' in Figures 12 and 13. T and T' connect terminal points $s_1, ..., s_6$ to source s_0. Both trees are assumed to extend to the right side of s_0, with a mirror image of the subtrees on the left of s_0. In this example, we set both the resistance and capacitance per unit length, α and β, to one, and the loading capacitance c_{L_i} of each sink node s_i to zero. [4]

ZST T was constructed so that if points s_1 and s_2 are merged at point p_1, then vertical wires from points s_3 through s_6 will merge along the horizontal wire from s_1 to s_0 with exactly zero skew. If s_1 and s_2 are merged on the merging segment between them, then the delay at p_1 will increase, and jogs will be required in the edges e_{s_3} through e_{s_6}. In this example, the four required jogs are each greater than 0.3. Hence their sum is greater than 1, the the amount of wire saved initially in T' by merging s_1 and s_2 at p_0.

Table 2 contains the calculated delay and capacitance at each of the interior nodes of T and T'. For example in T the capacitance at p_1, C_{p_1} is 33; and the delay at node p_2 is

$$t_{ED}(p_2) = t_{ED}(p_1) + 0.1 * (C_{p_1} + \frac{0.1}{2}) = 60.5 + 3.305 = 63.8 = \frac{(11.297)^2}{2}$$

The node capacitance at any interior node is defined to equal the cost of its subtree plus the sum of the loading capacitances of the subtree's leaves. Since all loading capacitances are zero in our example, the node capacitances equal the cost of the corresponding subtrees. Consequently, we see from Figure 2 that $cost(T)$ is less than $cost(T')$ by 0.44. Thus the embedding algorithm DME does not always produce a minimum cost ZST for a given topology.

[4] The example can be easily altered to include non-zero loading capacitances. To do this, shorten each edge adjacent to a terminal node by a small value $c > 0$ and and set the loading capacitance of all terminal nodes to c.

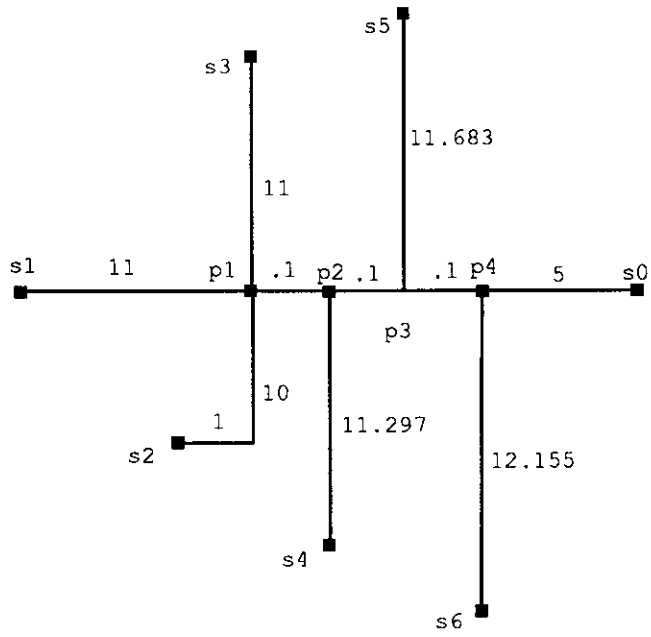

Figure 12: ZST T which violates the embedding algorithm but has optimal cost for its topology. Note that the tree is not drawn to scale and the lengths of segments are as indicated.

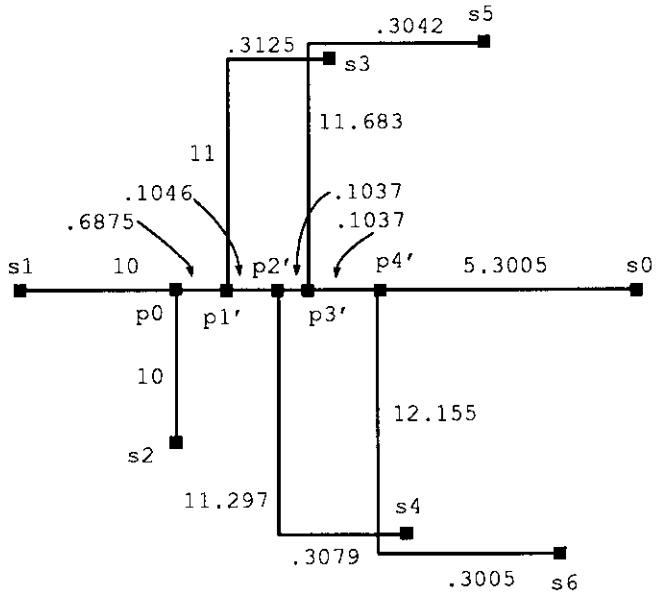

Figure 13: ZST T which violates the embedding algorithm but has optimal cost for its topology. Note that the tree is not drawn to scale and the lengths of segments are as indicated.

References

[1] H. Bakoglu, J. T. Walker and J. D. Meindl, "A Symmetric Clock-Distribution Tree and Optimized High- Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits", *Proc. IEEE Intl. Conf. on Computer Design* Port Chester, NY, October 1986, pp. 118-122.

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

Tree T			Tree T'		
node	delay	capacitance	node	delay	capacitance
			p_0	50	20
p_1	60.5	33.0	p'_1	64.0	32.0
p_2	63.8	44.4	p'_2	67.3	43.7
p_3	68.2	56.2	p'_3	71.9	55.8
p_4	73.9	68.4	p'_4	77.6	68.4
s_0	428.6	2×73.44	s_0	454.0	2×73.66

Table 2: Delay and capacitance at each internal node of ZSTs T and T'.

[3] K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees With Minimum Wirelength," to appear in *Proc. IEEE Intl. Conf. on ASIC*, 1992.

[4] S. Boon, S. Butler, R. Byrne, B. Setering, M. Casalanda and Al Scherf, "High Performance Clock Distribution For CMOS ASICS," *IEEE Custom Integrated Circuits Conference*, pp. 15.4.1-15.4.4, 1989.

[5] J. Burkis, "Clock Tree Synthesis for High Performance ASICs", *IEEE Intl. Conf. on ASIC*, pp. 9.8.1-9.8.4. 1991.

[6] S. Dhar, M. A. Franklin and D. F. Wann, "Reduction of Clock Delays in VLSI Structures," *Proc. IEEE Intl. Conf. on Computer Design*, 1984, pp. 778-783.

[7] M. Edahiro, "A Clock Net Reassignment Algorithm Using Voronoi Diagram," *IEEE Intl. Conf. on Computer-Aided Design*, pp. 420-423, Nov. 1990.

[8] W. C. Elmore, "The Transient Response of Damped Linear Networks With Particular Regard to Wide-Band Amplifiers," *Journal of Applied Physics*, vol.19, no.1, pp. 55-63, Jan. 1948.

[9] J. P. Fishburn, "Clock Skew Optimization," *IEEE Transactions on Computers*, vol. 39. No 7, pp. 945-951, July 1990.

[10] A. L. Fisher and H. T. Kung, "Synchronizing Large Systolic Arrays", *Proceedings of SPIE* 341, May 1982, pp. 44-52.

[11] M. Garey and D. S. Johnson, "The Rectilinear Steiner Problem is NP-Complete", *SIAM J. of Applied Math.* 32(4) (1977), pp. 826-834.

[12] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High Performance ICs," *27th ACM/IEEE Design Automation Conference*, pp. 573-579, 1990.

[13] A. B. Kahng, J. Cong, and G. Robins, "High-Performance Clock Routing Based on Recursive Geometric Matching," *28th ACM/IEEE Design Automation Conference*, pp. 322-327, 1991.

[14] P. Ramanathan and K. G. Shin, "A Clock Distribution Scheme for Non-Symmetric VLSI Circuits," *IEEE Intl. Conference on Computer-Aided Design*, pp. 398-401, Nov. 1989.

[15] 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.

[16] T. Sakurai, "Approximation of Wiring Delay in MOSFET LSI," *IEEE Journal of Solid-State Circuits* 18(4), August 1983, pp. 418-426.

[17] K. P. Shambrook, "An Overview of Multichip Module Technologies", *Proc. IEEE Workshop on Multichip Modules*, March 1991, pp. 1-6.

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

[19] D. F. Wann and M. A. Franklin, "Asynchronous and Clocked Control Structures for VLSI Based Interconnection Networks," *IEEE Transactions on Computers*, 21(3), March 1983, pp. 284-293.

Appendix A:

Summary of Work Submitted to IEEE Intl. Conf. on ASIC

Zero-Skew Clock Routing Trees With Minimum Wirelength

Kenneth Boese and Andrew Kahng
UCLA Computer Science Dept., Los Angeles, CA 90024-1596
Communicating Author: Andrew Kahng, CS Dept., 3732 Boelter Hall, UCLA,
Los Angeles, CA 90024-1596; Tel. 310-206-7073, FAX 310-825-2273

Technical Area: ASIC/MCM CAD Tools (Clock Tree Synthesis)

Abstract

Clock routing trees are an important component in the design of high performance VLSI systems. Two objectives in designing clock trees are to minimize clock skew and to minimize the total length of wire. We present an algorithm to construct a zero-skew clock routing tree that minimizes the amount of wire used, subject to compatibility with a prescribed connection topology. The algorithm improves upon previous algorithms that choose routing tree topologies but do not exploit certain properties of the Manhattan metric for wire placement. In fact, our work may be used to improve the clock trees constructed by previous algorithms in the literature; moreover, it extends to the Elmore delay model in addition to the linear delay model, and again results in zero-skew trees with reduced wirelength.

Technical Summary

We present an algorithm to design minimum-skew clock routing trees for high-performance VLSI systems. We represent a clock routing tree by its *connection topology*, which is an unweighted binary tree whose leaves are the terminals to be connected by the clock tree. The clock tree *skew* is the maximum difference in signal delay along any two source-terminal paths. Our objective is to construct a clock routing tree with zero skew, where delay may be computed by either the linear or Elmore delay [1] models.

The algorithm produces a zero-skew clock routing tree (ZCRT) in linear time via a two-phase approach: we employ bottom-up merging of subtrees with respect to a given topology, and then find optimal locations of the internal nodes during a subsequent top-down phase. In contrast, previous clock tree constructions (e.g., [3] [5]) use approaches such as bottom-up matching in order to define the clock tree topology; such methods suffer from a lack of flexibility in that they permanently embed each internal node of the tree as it is defined.

We have proved the following results. In the linear delay model, (i) our algorithm produces the minimum cost ZCRT, i.e., the tree that uses the minimum amount of wire, with respect to the prescribed connection topology; (ii) this tree will have the minimum possible source-terminal delay, equal to one-half the diameter of the set of clock terminals; and (iii) the algorithm finds the placement of the clock source for the ZCRT which will yield minimum total tree cost over all possible topologies. In the Elmore delay model, (iv) the algorithm applies techniques of [5] to compute edge lengths and produces a zero-skew tree that reduces total wirelength via effective choice location of "tapping" points. Moreover, the algorithm may be used to derive the minimum-cost trees compatible with connection topologies constructed by other algorithms, e.g., [2] [3] [5]. Because the procedure is bottom up and iterative, it can readily be applied to multi-staged clock trees or other hierarchical systems. Finally, with either the linear or the Elmore delay model, the algorithm runs in linear time; this surprising efficiency seems to be an artifact of the Manhattan metric[1] In practice, the linear-time complexity will be dominated by the time required to compute a heuristic connection topology.

[1] Note that a more general linear programming approach may be applicable to perform optimal "Steiner point shifting" for a given topology; such a method may succeed in alternate geometries to which our algorithm does not apply.

Appendix: Details of the Algorithm (Linear Delay Model)

The input to algorithm ALG1 will be a point set $P \subseteq \Re^2$ and topology G for connecting P. The output will be a ZCRT T that has minimum cost over all embeddings in $\Re^2$ of the topology G.

Phase I constructs, in a bottom up manner, the intersections of diamond-shaped regions surrounding each of the terminal points in P. The intersection of any number of diamond-shaped regions will be rectangular regions tilted at a 45-degree angle from the x- and y-axes. The intersections are performed in the order determined by topology G and the centers or "cores" of each intersection will contain possible placements for the corresponding interior node.

Phase II works in a top-down manner to find the exact placement of interior nodes. It places the source of the clock tree at any point on the core constructed in Phase I for the root node. For every other interior node of the topology, Phase II selects any point from the corresponding core that is within a prescribed distance (which we call ec in the discussion below) of the embedding of its parent node.

Definition: A *tilted rectangular region* or *TRR* is the union of a rectangle and its interior region where the sides of the rectangle all have slope 1 or -1. For consistency, we say that line segments having slope 1 or -1, single points, and the empty set are all also TRR's.

Define the **core** of a TRR to be the subset of the TRR that is of maximum Manhattan distance from its perimeter. The core will always be a line segment, a point, or the empty set (when the TRR is itself empty). The **radius** of a TRR is the distance from its core to its perimeter. For instance, the TRR with corners (0,1), (1,0), (2,3), and (3,2) has radius 1; its core is the line segment with endpoints (1,1) and (2,2). For any TRR R, we use $core(R)$ and $radius(R)$ to respectively denote the core and radius of R.

The details of our algorithm are as follows:

- (Initialization) For the point set P, let each $i \in P$ have coordinates (x_i, y_i). Calculate the diameter d of P as the maximum of

$$\max_{i \in P}(x_i - y_i) - \min_{i \in P}(x_i - y_i)$$

  and

$$\max_{i \in P}(x_i + y_i) - \min_{i \in P}(x_i + y_i).$$

  Construct a diamond-shaped TRR with radius $d/2$ around each point in P. Associate each diamond with its corresponding leaf node in the topology G. (The intersection of all of these TRR's is guaranteed to be non-empty.)

- (Phase I) In a bottom-up manner, construct a TRR for each interior node in G as the intersection of the TRR's of its two children. (The core of this TRR is a set of possible placements of the node.)

- (Phase IIa) At the root node, choose any point in the core of its TRR as the placement for the source node.

- (Phase IIb) In a top-down manner, choose actual placements of the interior nodes. For any parent and child node, let R_1 be the TRR of the parent and R_2 be the TRR of the child. The cost of the new edge will be sett to $radius(R_2) - radius(R_1)$. Call this value ec. Construct a diamond of radius ec around the embedding chosen for the parent node and then find the intersection of the core of R_2 with this diamond. Choose any point q in this intersection as the placement of the child node.

References

[1] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers", *J. Appl. Phys.* 19(1) (1948), pp. 55-63.

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

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

[4] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, New York, Springer-Verlag, 1985.

[5] R. S. Tsay, "Exact Zero Skew", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 336-339.