Zero Skew Clock Routing With Minimum Wirelength*

Ting-Hai Chao†, Yu-Chin Hsu‡, Jan-Ming Ho¶,
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. 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 first present the Deferred-Merge Embedding (DME) algorithm, which embeds any given connection topology to create a clock tree with zero skew while minimizing total wirelength. The algorithm always yields exact zero skew trees with respect to the appropriate delay model. Experimental results show an 8% to 15% wirelength reduction over previous constructions in [17] [18]. The DME algorithm may be applied to either the Elmore or linear delay model, and yields *optimal* total wirelength for linear delay. DME is a very fast algorithm, running in time linear in the number of synchronizing elements. We also present a unified BB+DME algorithm, which constructs a clock tree topology using a top-down *balanced bipartition* (BB) approach, and then applies DME to that topology. Our experimental results indicate that both the topology generation and embedding components of our methodology are necessary for effective clock tree construction. The BB+DME method averages 15% wirelength savings over the previous method of [17], and also gives 10% average wirelength savings when compared to the method of [25]. The paper concludes with a number of extensions and directions for future research.

1 Introduction

In synchronous VLSI designs, circuit speed is increasingly limited by two factors: (i) delay on the longest path through combinational logic, and (ii) clock skew, which is the maximum difference in arrival times of the clocking signal at the synchronizing elements of the design. This is seen from the following well-known inequality governing the clock period of a clock signal net [2] [17]:

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

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

*The work of A. B. Kahng and K. D. Boese was supported in part by NSF MIP-9110696, ARO DAAK-70-92-K-0001, ARO DAAL-03-92-G-0050, and a GTE Graduate Fellowship. A. B. Kahng is also supported by an NSF Young Investigator Award.
Author affiliations: (§) Dept. of Computer Science, UCLA, Los Angeles, CA 90024-1596; (†) Computer and Communication Research Laboratories, ITRI, Hsin-Chu, Taiwan 31015 R.O.C.; (‡) Dept. of Computer Science, UC Riverside, Riverside, CA 92521; (¶) Institute of Information Science, Academia Sinica, Taipei, Taiwan 11529 R.O.C.

switching speeds due to advances in VLSI fabrication technology will significantly decrease the terms t_{su}, t_{ds}, and t_{d_gates}. Therefore, $t_{d\_interconnect}$ and t_{skew} become the 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.

Several results address formulations with inherently small problem size. For building block design styles, Ramananathan and Shin [21] have 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] have also proposed hierarchical clock tree synthesis approaches involving geometric clustering and buffer optimization at each level. More powerful clock tree resynthesis or reassignment methods were used by Fishburn [13] and Edahiro [11] to minimize the clock period while avoiding hazards or race conditions; Fishburn employed a mathematical programming formulation, while Edahiro employed a clustering-based heuristic augmented by techniques from computational geometry. All of these methods are essentially limited to small problem sizes, either by their algorithmic complexity or by their reliance on strong hierarchical clustering. In contrast, we are interested in clock tree synthesis for "flat" problem instances with many sinks (synchronizing elements), as will arise in large standard-cell, sea-of-gates, and multichip module designs.

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] [10] [14] [26]. The H-tree structure can significantly reduce clock skew [10] [26], but is applicable only when all of the sinks have identical loading capacitances and are placed in a symmetric array. A more robust clock tree construction for cell-based layouts is due to Jackson, Srinivasan and Kuh [17]: their "method of means and medians" (MMM) algorithm generates a topology by recursively partitioning the set of sinks into two equal-sized subsets, then connecting the center of mass of the entire set to the centers of mass of the two subsets. While the MMM solution will have reasonable skew on average, Kahng et al. [18] gave small examples for which the source-sink pathlengths in the MMM solution may vary by as much as half of the chip diameter. In some sense, this reflects an inherent weakness in the top-down approach: it can commit to an unfortunate topology early on in the construction. Kahng et al. [18] [9] have proposed a bottom-up matching approach to clock tree construction: in practice their method eliminates all source-sink pathlength skew, while using 5%-7% less total wirelength than the MMM algorithm. However, as the method of [18] [9] focuses primarily on pathlength balancing, their method addresses clock skew minimization only in the sense of the *linear* delay model. Tsay [25] uses ideas similar to both [17] and [18], and achieves exact zero skew trees with respect to the Elmore delay model [12] [22]. His algorithm was the first to produce trees with exact zero skew in all cases. In the same spirit as the method of [18], Tsay's method recursively combines pairs of zero skew trees at "tapping points", analogous to the "balance points" in [18], to yield larger zero skew trees.

The primary motivation behind our work is to minimize the total wirelength of clock routing trees while maintaining exact zero skew with respect to the appropriate delay model. Total wirelength is a critical parameter of the clock routing solution since excess interconnect not only increases layout area but also results in greater tree capacitance, thus requiring more power for distribution of the clock signal. However, both the top-down method of [17] and the bottom-up methods of [18] [9] [25] 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. These previous methods are actually quite inflexible in that they permanently embed each internal node of the tree as soon as it becomes defined [18], or else choose the embedding with at most one level of lookahead in the tree construction [17] [25].

In this paper, we first 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 4 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 15% over the MMM algorithm [17] and 8% over the method of Kahng et al. [18]. In all cases, our clock trees have *exact* zero skew according to the appropriate delay model, and our Elmore delay computations have been confirmed by SPICE simulations which show sub-picosecond skew on all benchmark examples.

Since the DME algorithm only optimizes a prescribed topology, it cannot achieve all possible improvement of the clock tree construction. Thus, to complement this successful embedding method, we also propose a new top-down heuristic for constructing an initial clock tree topology, based on the geometric concept of a *balanced bipartition* (BB). Applying our embedding to topologies generated in this way yields a unified BB+DME algorithm which gives very promising results: we achieve 15% reduction in tree cost and as compared with the MMM algorithm [17], and we achieve 10% reduction in tree cost and a 22% reduction in Elmore delay as compared with the method of Tsay [25].[1] Again, all of our solutions have exact zero skew. 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. Furthermore, because our method implicitly maintains *all* possible minimum-cost embeddings of a topology, it may be used to reroute the clock net while preserving minimum wirelength, as may be necessary when channel density must be

[1]Note that SPICE simulations for BB+DME constructions on random sink sets (Table 4 below) indicate only a 3% improvement in delay compared to the MMM algorithm. This suggests that although the Elmore model is reasonably accurate for predicting skew, it is less accurate for predicting delay.

minimized.

The remainder of this paper 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; (ii) application of the DME algorithm to both the linear and Elmore delay regimes; and (iii) our unified BB+DME algorithm, which uses a top-down balanced bipartitioning (BB) strategy to derive a good tree topology to which the DME algorithm may be applied. Section 4 gives experimental results and comparisons with previous work, and Section 5 concludes with directions for future research.

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.[2] The embedding associates a *placement* in $\Re^2$ with each node $v \in G$; we will use $pl(T, v)$ or $pl(v)$ to represent this location. (When no confusion arises, we may also denote $pl(T, v)$ simply by v.) 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 written as 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_w is denoted by $|e_w|$, and must be greater than or equal to the Manhattan distance between its endpoints $pl(w)$ and $pl(v)$.[3] The cost of $T(S)$, denoted $cost(T(S))$, 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 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 the zero skew clock routing problem asks for a minimum cost ZST with a prescribed connection topology:

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

[2] Note that the binary tree representation suffices to capture arbitrary Steiner routing topologies. Also, because the meaning is clear, we use $T(S)$ instead of $T(S, G)$ to denote a clock tree; implicitly, the embedding is always with respect to a particular topology G.

[3] To route a wire of greater length than the distance between its endpoints, the method of specified-length routing due to Hanafusa et al. [16] can be used.

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.[4] Using equations such as those of Rubinstein et al. [22], one can achieve tight upper and lower bounds on delay in a distributed RC tree model of the clock net. However, in practice it is appropriate to apply one of two simpler RC delay approximations, either the the linear model or the Elmore model, both of which are easier to compute and optimize during clock tree design.

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 rest of the connection topology. Normalized by an appropriate constant factor, the linear delay between any two nodes u and w in a source-sink path is

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

While less accurate than the distributed RC tree delay formulas of Rubinstein et al [22], the linear delay model has been effectively used in clock tree synthesis [18] [21]. In general, use of the linear approximation is reasonable with older ASIC technologies, which have larger mask geometries and slower packages. Tsay [25] notes that the linear delay model is also proper for emerging optical and wave interconnect technologies. In addition, we observe that linear delay applies to hybrid packaging technologies, which have relatively large interconnect geometries [24].

2.1.2 Elmore Delay

With smaller device dimensions and higher ASIC system speeds, a distributed RC tree model for signal delay in clock nets is often required to derive accurate timing information. Typically, we use the first-order moment of the impulse response, also known as the Elmore delay [6] [8] [25]. The Elmore delay model is developed as follows. 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 edge e_v are given by $\alpha \cdot |e_v|$ and $\beta \cdot |e_v|$, respectively. 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 $T(S)$ rooted at v, and let c_v denote the node capacitance of v.[5] The

[4] 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.

[5] As noted earlier, we will assume that $c_v = 0$ for each internal node in all of our examples and benchmarks.

tree capacitance of T_v is denoted by C_v and equals the sum of capacitances in T_v. C_v is calculated using the following recursive formula:

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

According to [12] [22] [23], the Elmore delay $t_{ED}(s_0, s_i)$ can be calculated by the following formula (see [25] for a discussion of underlying circuit models):

$$t_{ED}(s_0, s_i) = \sum_{e_v \in path(s_0, s_i)} r_{e_v}(\frac{1}{2} c_{e_v} + C_v).$$

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

$$t_{ED}(u, w) = \sum_{e_v \in path(u, w)} r_{e_v}(\frac{1}{2} c_{e_v} + C_v).$$

Elmore delay is additive: if v is a vertex on the u-w path, then $t_{ED}(u, w) = t_{ED}(u, v) + t_{ED}(v, w)$, and in particular, if v is a child of u on the u-s_i path, then $t_{ED}(u, s_i) = r_{e_v}(\frac{1}{2} c_{e_v} + C_v) + t_{ED}(v, s_i)$. A sink node s_i may be treated as a trivial zero skew subtree with capacitance c_{L_i} and delay zero.

3 Main Results

This section presents our new unified approach to constructing a ZST over a given set of sinks S. At a high level, we divide the construction of the ZST into: (i) generation of a connection topology, and (ii) embedding of that connection topology in the Manhattan plane. Our discussion begins with the Deferred-Merge Embedding (DME) algorithm, which computes a wire-efficient embedding of a given topology. Next, we describe the application of the DME algorithm to both the linear and Elmore delay models. We then present a new top-down *balanced bipartition* (BB) algorithm that creates a good connection topology, leading to the unified BB+DME algorithm.

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

3.1.1 Bottom-Up Phase: The Tree of Merging Segments

For prescribed sink locations S and connection topology G, we construct a tree of *merging segments*. The basic idea is as follows. Each node v in G, is associated with a merging segment which represents a set of possible placements of v. The merging segment of a node depends on the merging segments of its two children, so the connection 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 $|e_b|$ denote the lengths to be assigned to edges e_a and e_b. These lengths are chosen to minimize merging cost while balancing delays at $pl(v)$. Because delay is a monotone increasing function of wirelength, there is a unique optimal assignment of lengths to e_a and e_b.[6]

We now develop more precisely the construction of the tree of merging segments. A *Manhattan arc* is defined to be 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 Manhattan arc at the center of the TRR is called its *core*. The *radius* of a TRR is the distance between its core and its boundary.

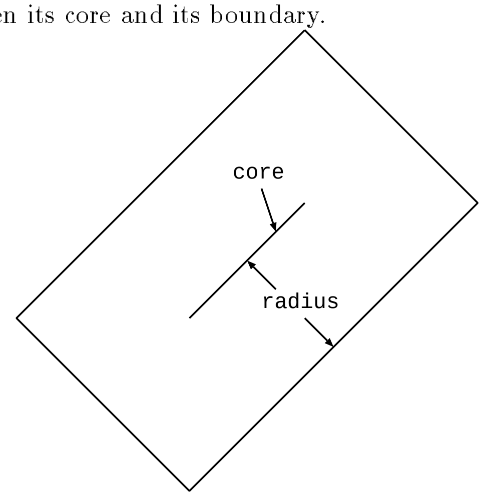

Figure 1: An example of a TRR with core and radius as indicated.

The *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 placements $pl(v)$ which allow minimum merging

[6] The uniqueness is shown as follows. Suppose the minimum merging cost is c. Define a function $f(|e_a|)$ to be the path delay from v to sinks in TS_a for edge length $|e_a|$; similarly define $g(|e_b|)$ for the path delay from v to sinks in TS_b. Define $g'(|e_a|) = g(c - |e_a|)$. A length assignment to e_a must satisfy $f(|e_a|) = g'(|e_a|)$, or alternatively, $(f - g')(|e_a|) = 0$. If both f and g are monotone increasing functions, then g' is monotone decreasing and $f - g'$ is monotone increasing. Thus $(f - g')(|e_a|) = 0$ will have at most one solution.

cost, that is to say, 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., $ms(v) = trr_a \cap trr_b$.

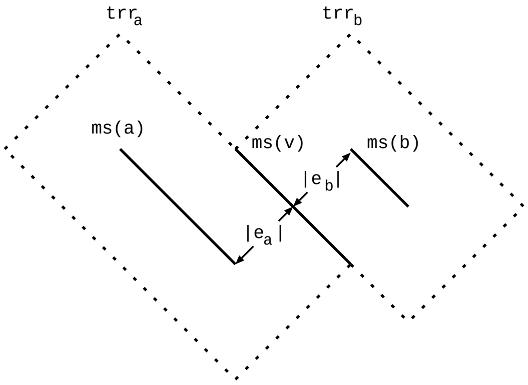

Figure 2: Construction of merging segment $ms(v)$ when the merging cost equals κ.

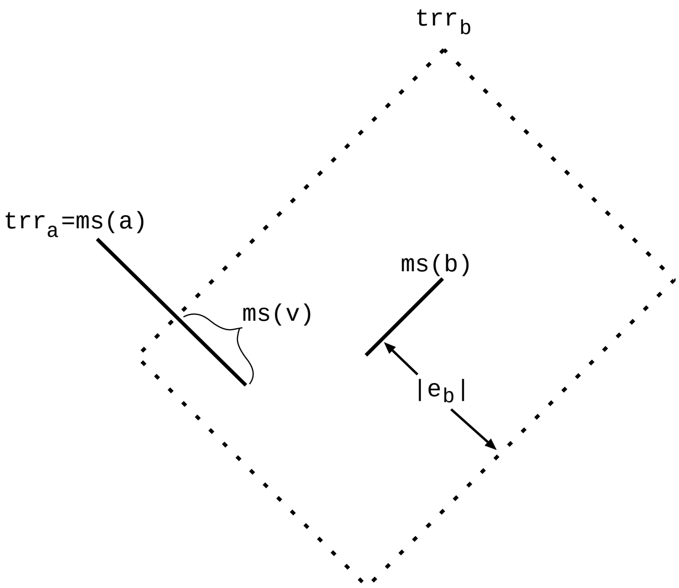

Figure 3: Construction of merging segment $ms(v)$ when the merging cost is greater than κ. Note that in this example, $radius(trr_a) = |e_a| = 0$.

The merging cost at v has an obvious lower bound of $\kappa = d(ms(a), ms(b))$. If the merging cost is greater than κ (i.e., more wirelength is needed to balance the delays), then one edge length will equal zero and the other will equal the merging cost. Figure 2 illustrates the algorithm for the case where the merging cost is equal to κ, and Figure 3 illustrates the algorithm for the case where the merging cost is greater than κ. An entire tree of merging segments is illustrated by Figure 4. The leaves of the tree of segments are all single points representing the sink locations $s_1, \ldots, s_8$, and the internal nodes are Manhattan arcs.

We prove that all merging segments are Manhattan arcs using induction and the following lemma. (Proofs of all lemmas are given in the Appendix.)

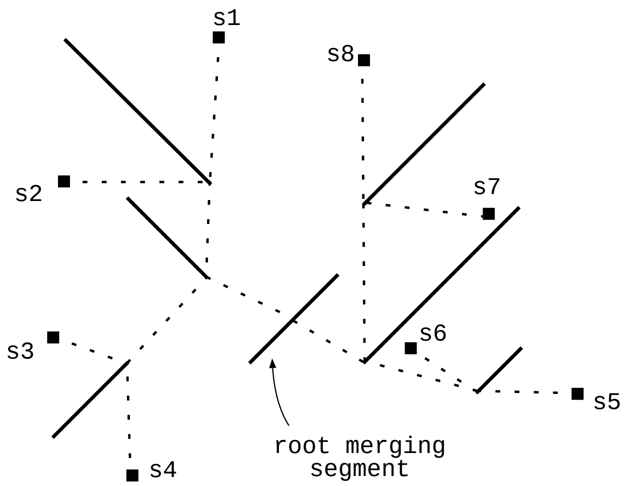

Figure 4: An example of a tree of merging segments with sinks $s_1, \ldots, s_8$. The solid lines are merging segments and the dotted lines indicate edges between merging segments.

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 the TRR $R_1 \cap R_2$ is also a Manhattan arc.*

Lemma 1 implies that if $ms(a)$ and $ms(b)$ are both Manhattan arcs, then $ms(v)$ is a Manhattan arc, as follows: (i) if the merging cost at v is equal to κ, then $d(core(trr_a), core(trr_b)) = |e_a| + |e_b| = radius(trr_a) + radius(trr_b)$, and hence, $trr_a \cap trr_b$ is a Manhattan arc; or ii) if the merging cost at v is greater than κ, then either trr_a or trr_b will be a Manhattan arc whose intersection with any convex set will also be a Manhattan arc. For each sink s_i, the merging segment $ms(s_i)$ is a single point and thus a Manhattan arc. By induction, therefore, all merging segments must be Manhattan arcs.

Procedure Build_Tree_of_Segments

Input: Topology G; set of sink locations S

Output: Tree of merging segments TS containing $ms(v)$ for each node v in G and edge length $|e_v|$ 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 5: Construction of the tree of segments.

Figure 5 gives a precise description of the procedure Build_Tree_of_Segments, which constructs the tree of merging segments. Details of the Calculate_Edge_Lengths subroutine depend on the delay model and are described in Sections 3.2.1 and 3.3.1 below.

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

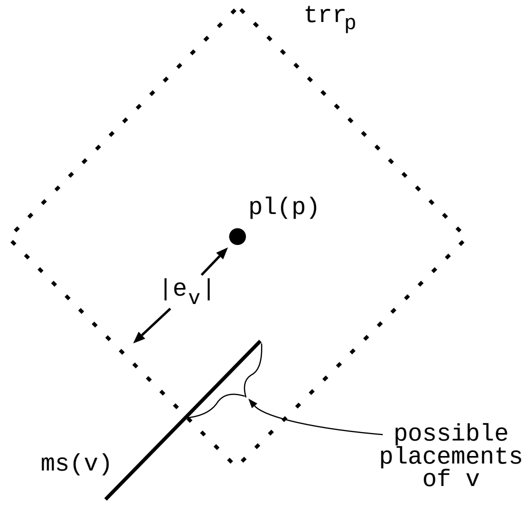

Figure 6: Procedure Find_Exact_Placements: finding the placement of v given the placement of its parent p.

```
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 pl(v) ∈ ms(v)
    else
        Let p be the parent node of v
        Construct trr_p as follows:
            core(trr_p) ← {pl(p)}
            radius(trr_p) ← |e_v|
        Choose any pl(v) ∈ ms(v) ∩ trr_p
    endif
```

Figure 7: Construction of the ZST by embedding internal nodes of the topology.

3.1.2 Top-Down Phase: 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, (i) if v is the root node, then select any point in $ms(v)$ to be $pl(v)$;[7] or (ii) if v is an internal node other than the root, choose $pl(v)$ to be any point in $ms(v)$ that is at distance $|e_v|$ or less from the placement of v's parent p (because the merging segment $ms(p)$ was constructed such that $d(ms(v), ms(p)) \leq |e_v|$, there must exist some choice of $pl(v)$ satisfying this condition). In case (ii), the algorithm first creates a square TRR trr_p with radius $|e_v|$ and core equal to $\{pl(p)\}$; then, $pl(v)$ can be any point from $ms(v) \cap trr_p$ (see Figure 6). For the tree of merging segments in Figure 4, the resulting placements are indicated by the points at which the segments are connected by dotted lines. Figure 7 describes the procedure Find_Exact_Placements, which uses the tree of merging segments to determine the final embedding of nodes in the ZST.

The time complexity of DME is analyzed as follows. Because each instruction in Find_Exact_Placements is executed at most once for each node in G (and the intersection of TRRs $ms(v)$ and trr_p can be found in constant time by Lemma 1), Find_Exact_Placements runs in time linear in the size of S. Because procedure Build_Tree_of_Segments also runs in linear time, DME as a whole is a linear-time algorithm.

3.2 Application of DME to Linear Delay

3.2.1 Calculating Edge Lengths

Calculating the edge lengths $|e_a|$ and $|e_b|$ is straightforward in the linear delay model. Let a and b be children of v with merging segments $ms(a)$ and $ms(b)$, and let $t_{LD}(a)$ and $t_{LD}(b)$ be the delays from a and b to the sinks in their respective subtrees. Then, zero skew at v requires that

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

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

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

and

$$|e_b| = \kappa - |e_a|.$$

On the other hand, if $|t_{LD}(a) - t_{LD}(b)| > \kappa$, then the merging cost is minimized when one of the edge lengths is equal to zero. It is easy to see that 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 $|e_b| = 0$ and $|e_a| = t_{LD}(b) - t_{LD}(a)$.

[7] If a fixed source location s_0' is specified, choose $pl(s_0) \in ms(s_0)$ with minimum distance from s_0' and connect a wire directly from s_0' to $pl(s_0)$.

3.2.2 Optimality of DME for Linear Delay

The following theorem states that the DME algorithm is optimal in the linear delay regime.

Theorem 1 *Given a set of sink locations S and a connection topology G, the DME algorithm produces a ZST T with minimum cost over all ZSTs for S having topology G.*

The proof of Theorem 1 relies on Lemmas 2 and 3. Lemma 2 asserts that for any node v in an optimal ZST, $pl(v)$ is in $ms(v)$ and must therefore satisfy the constraints imposed in the bottom-up phase of the algorithm. Lemma 3 implies that the placements of two sibling nodes correspond to a closest pair of points in their respective merging segments. Together, Lemmas 2 and 3 can be used to show that placements in an optimal ZST must satisfy the top-down phase of the algorithm. Let $t_{LD}(T, x)$ denote the delay in ZST T between a point x in T and each sink which has x on its source-sink path.

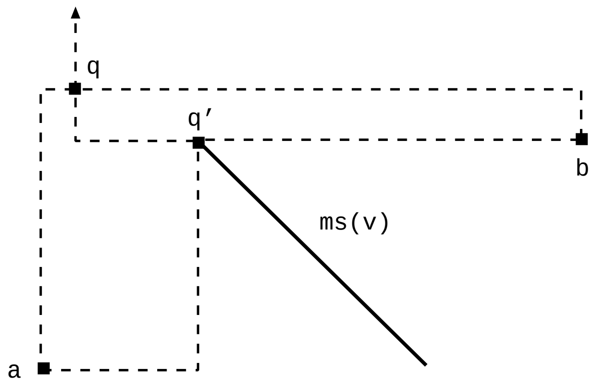

Figure 8: Optimal placement of v must be on $ms(v)$. $pl(T, v) = q$; $pl(T', v) = q'$; and $cost(T') < cost(T)$.

Lemma 2 : *Given a ZST T with topology G, let v be an internal node with children a and b. Suppose the subtrees of T rooted at a and b can be generated by the DME algorithm for some placement of v on $ms(v)$, and also suppose that $q = pl(T, v) \notin ms(v)$. Then a new ZST T' with the same topology can be constructed from T by moving the placement of v so that the following hold: (i) $q' = pl(T', v) \in ms(v)$; (ii) $cost(T') < cost(T)$; and (iii) $t_{LD}(T, q) = t_{LD}(T', q)$.*

Lemma 2 is illustrated in Figure 8. The construction of T' from T reduces the tree cost by modifying the q–a and q–b connections so that they share wire on the segment from q to q'.

Lemma 3 : *Suppose that a and b are two sibling nodes in ZST T with parent v, and suppose 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 ZST T' can be constructed from the same topology, with $cost(T') < cost(T)$ and with $t_{LD}(T, q) = t_{LD}(T', q)$ for $q = pl(T, v)$.*

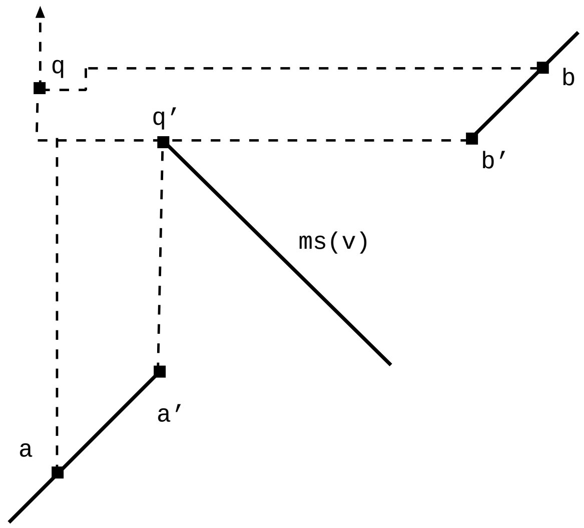

Figure 9: Optimal placement of siblings a and b must satisfy the distance constraint in the top-down phase Find_Exact_Placements. $pl(T,a) = a$ and $pl(T',a) = a'$, etc.; and $cost(T') < cost(T)$.

Figure 9 contains an illustration of Lemma 3. Moving the placements of nodes a and b to locations a' and b' allows the a'–q and b'–q connections to share wire on the segment from q' to q. The delay at point q remains unchanged.

Proof of Theorem 1: The proof is by contradiction. The DME algorithm places only two constraints on the placement of a node v in G: (i) $pl(v) \in ms(v)$ and (ii) $d(pl(v), pl(p)) \leq L_v$, where p is the parent of v and L_v is the edge length assigned by DME to e_v. Condition (i) arises by the construction in the top-down phase of DME, and condition (ii) is required by the bottom-up phase of DME. Suppose ZST T has minimum cost for point set S and topology G, but contains a node placement violating one of the two conditions. Let v be a node with greatest depth in T that violates either condition, and let w be the sibling of v. Because v has maximum depth, all of the descendants of v and w can be produced using DME. Consequently, because T has minimum cost, Lemma 2 implies that $pl(T,v)$ must be in $ms(v)$ and $pl(T,w)$ must be in $ms(w)$. Thus, v does not violate condition (i).

Consequently, v must violate (ii), i.e., $d(pl(T,v), pl(T,p)) > L_v$. Let $L(T,e_v)$ denote the length of edge e_v in T. Because the length of an edge must be at least the distance between its endpoints, $L(T,e_v) > L_v$. Suppose $d(pl(T,v), pl(T,w)) \leq d(ms(v), ms(w))$. Then the subtrees of T rooted at v and w can be generated by DME for some placement of p on $ms(p)$, and by Lemma 2, $cost(T)$ can be improved by moving p to its merging segment and setting $L(T',e_v) = L_v$ and $L(T',e_w) = L_w$. If $d(pl(T,v), pl(T,w)) \leq |t_{LD}(v) - t_{LD}(w)|$, then $cost(T)$ can be reduced by moving $pl(p)$ to $pl(v)$ if $L_v = 0$, or to $pl(w)$ if $L_w = 0$. Hence, we must have $d(pl(T,v), pl(T,w)) > d(ms(v), ms(w))$, and $d(pl(T,v), pl(T,w)) > |t_{LD}(v) - t_{LD}(w)|$. Then by Lemma 3 $cost(T)$ can be decreased, contradicting the assumption that T has minimum cost. $\square$

It can be proved that in the linear model, DME also minimizes the source-sink delay in a ZST, and that this delay is equal to one-half the diameter of the sink set S. A proof of this result is contained in [3].

The DME algorithm is also optimal for any topology in the variant of the ZST problem where the source location is pre-defined. Suppose that $ms(s_0)$ is the merging segment for the root node s_0 of topology G and that s_0' is the prescribed source location. The DME algorithm can be modified at the beginning of the procedure Find_Exact_Placements to connect s_0' with the closest point in $ms(s_0)$. This point becomes $pl(s_0)$. Lemmas 2 and 3 can be used to prove the optimality of this method: they state that any tree rooted at a location $q \notin ms(s_0)$ will have minimum cost only if the two subtrees of G directly below the root are merged at a point $q' \in ms(s_0)$ which is then connected to s_0' by a single edge.

3.3 Application to Elmore Delay

3.3.1 Calculating Edge Lengths in the Elmore Delay Model

To calculate the edge lengths needed to merge two trees of merging segments TS_a and TS_b with minimum merging cost in the Elmore model, we use the analysis of Tsay [25]. Let TS_a and TS_b respectively have capacitance C_1 and C_2 and delay $t_1 = t_{ED}(a)$ and $t_2 = t_{ED}(b)$, and let $pl(v)$ be a merging point with minimum merging cost.

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

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

Let $d(ms(a), ms(b)) = \kappa$. Suppose that TS_a and TS_b can be merged with merging cost κ; in other words, $|e_a| = x$ and $|e_b| = \kappa - x$ for $0 \le x \le \kappa$. Then we have resistances $r_{e_a} = \alpha x$ and $r_{e_b} = \alpha(\kappa - x)$ and capacitances $c_{e_a} = \beta x$ and $c_{e_b} = \beta(\kappa - x)$. Substituting into (1) and solving for x yields

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

Case 1: If $0 \le x \le \kappa$, then there exists a feasible zero skew merging point of TS_a and TS_b with merging cost κ, $|e_a| = x$ and $|e_b| = \kappa - x$.

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 $\kappa' > \kappa$ is required to balance the delays of the two trees. If $x < 0$, which means $t_1 > t_2$, we choose $pl(a)$ as the merging point and set $|e_a| = 0$ and $|e_b| = \kappa'$. Then

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

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

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

Similarly, if $x > \kappa$, we set $|e_b| = 0$ and

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

The above analysis shows that a zero skew merging point between two ZSTs can always be found. The merging cost depends on the distance between the two roots of the ZSTs, the delay of each ZST, and the tree capacitance of each ZST. Intuitively, to minimize the merging cost we should therefore choose topologies such that merged subtrees have minimum distance between their roots, along with similar capacitances and delays, so as to avoid the extra cost $\kappa' - \kappa$. This motivates our new BB algorithm, which uses the geometric notion of a *balanced bipartition* for computing a topology. Before describing this algorithm in Section 3.4 below, we observe that the DME algorithm is not optimal for all topologies in the Elmore delay approximation model.

3.3.2 Suboptimality of DME for Elmore Delay

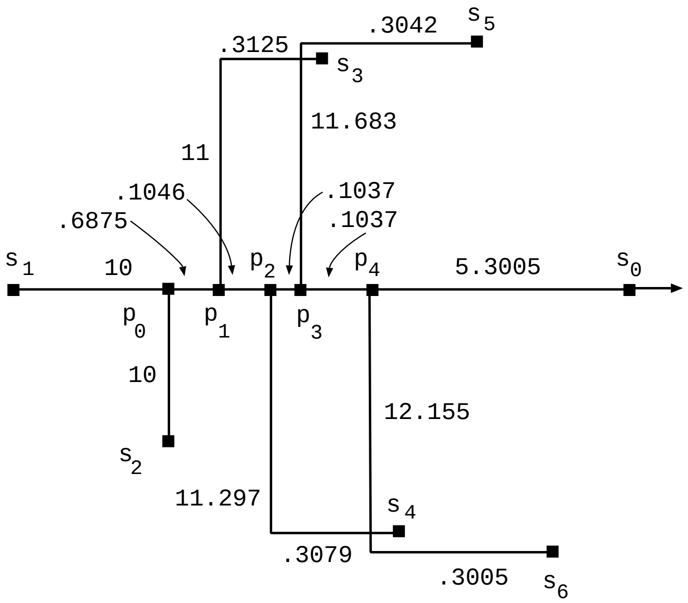

Figure 10: ZST T, which would be constructed by the DME algorithm with sub-optimal cost for its topology. (Note that the tree is not drawn to scale; lengths of horizontal and vertical segments are as indicated.)

Recall that in the linear delay regime, the DME algorithm produces an *optimum* (minimum wirelength) ZST for *any* given topology. Our experimental results in Section 4 clearly show the effectiveness of the DME algorithm in the Elmore delay model, and indeed we believe that in practice the algorithm gives solutions that are very close to optimum. However, the ZSTs T in Figure 10 and T' in Figure 11 demonstrate that, for some sink sets and topologies, DME will not be optimal for Elmore delay. 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 their subtrees on the right of s_0 being mirror images of the subtrees to the left of s_0 (this ensures that the source will be at s_0 in the optimal tree). In this example, we set both the unit resistance α and unit capacitance β to one, and the loading capacitance c_{L_s} of each sink node s to zero.[8]

The ZST T' in Figure 11 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, however, s_1 and s_2 are merged on their merging segment as shown in the tree T of Figure 10, 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 of length greater than 0.3. Thus, their sum is greater than 1, which was the amount of wire saved initially by merging s_1 and s_2 at p_0.

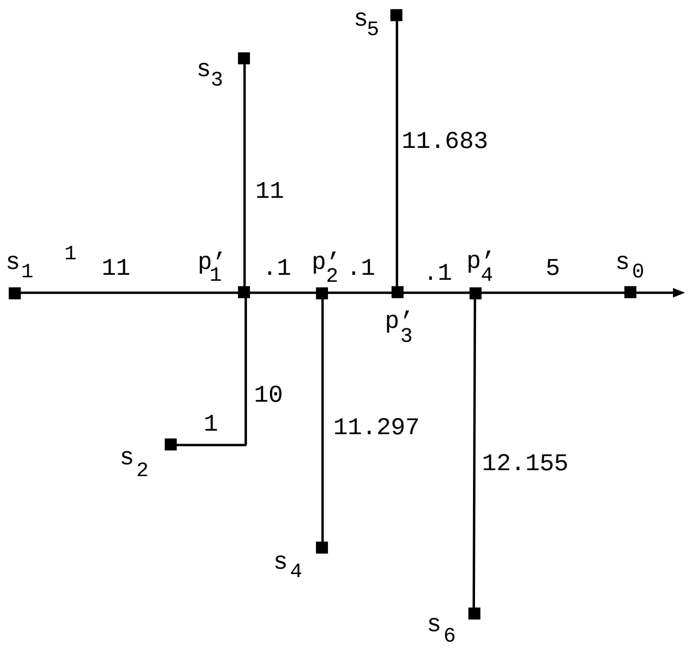

Figure 11: ZST T', which has optimal cost for the topology in Figure 10, but which violates the DME algorithm. In T', the internal nodes placed at p_0 and p_1 in T are placed at the same point, p'_1. (The tree is not drawn to scale; lengths of horizontal and vertical segments are as indicated.)

Table 1 contains the calculated delay and capacitance at each of the internal 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

$$t_{ED}(p'_2) = t_{ED}(p'_1) + 0.1 * (\frac{0.1}{2} + C_{p'_1}) = 60.5 + 3.305 = 63.8 = \frac{(11.297)^2}{2}$$

Because unit resistance and capacitance both equal one, and because loading capacitances at the leaves are zero, the tree capacitance of each node equals the amount of wire in its subtree. Thus, we see in Table 1 that $cost(T')$ is less than $cost(T)$ by 0.44.

[8]The example can be easily altered to have non-zero loading capacitances: shorten each edge adjacent to a terminal node by a small value $c > 0$, and then set the loading capacitance of each terminal node to c.

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

Table 1: Delay and capacitance at each internal node in ZSTs T and T'.

3.4 Topology Generation

It is easy to see that, as hinted by the examples of Figures 10 and 11, the choice of topology will affect the success of the DME embedding. We now present a new heuristic for generating connection topologies.[9] The heuristic works in top-down fashion, dividing the sink nodes recursively into two partitions with nearly equal total loading capacitance. We call this heuristic the Balanced Bipartition (BB) method. The BB method offers a more powerful top-down partitioning scheme than the previous approaches of Jackson et al. [17] and Tsay [25], which divide the sink set recursively, using only alternating horizontal and vertical cuts.

For our description of the BB method, we introduce the following notation. Denote the diameter of S by $dia(S) = max\{d(p,q) \mid p,q \in S\}$ and the number of sinks in S by $|S|$. Since the cost of any routing tree of S is greater than $dia(S)$ and less than $|S| \cdot \frac{dia(S)}{2}$, we consider $dia(S)$ to be a heuristic approximation of the cost of any ZST $T(S)$. Recall also that imbalanced loading capacitance may lead to excess edge length in the DME construction; we call a bipartition of a set of sinks S into two subsets S_1 and S_2 a *balanced bipartition* if the difference between the total loading capacitances of the two subsets is at most $max\{c_{L_i}\}$.[10] Intuitively, we would like to find a balanced bipartition which divides set S with minimum *partition cost*, given by $dia(S_1) + dia(S_2)$. This is the idea behind the BB heuristic. In the Euclidean metric, the problem of constructing a balanced bipartition which minimizes the sum of diameters can be solved in $O(n^2)$ time [19]. However, we are not aware of any polynomial-time algorithm that yields a minimum cost balanced bipartition in the Manhattan plane.

Let $p.x$ and $p.y$ be the x- and y-coordinates of point p. The *octagon* of set S is defined as the region formed by the intersection of eight half spaces (in clock-wise order around the octagon): $y \le \max_{p\in S}\{p.y\}$, $y - x \ge \min_{p\in S}\{p.y - p.x\}$, $x \ge \min_{p\in S}\{p.x\}$, $y + x \ge \min_{p\in S}\{p.y + p.x\}$, $y \ge \min_{p\in S}\{p.y\}$, $y - x \le \max_{p\in S}\{p.y - p.x\}$,

[9] No NP-completeness result has been obtained for our general minimum-cost zero skew clock tree formulation (i.e., where the topology has not been prescribed). However, [18] [9] showed that a closely related problem (in the linear delay model), the "bounded-skew pathlength-balanced tree problem", is trivially NP-complete since it reduces the minimum rectilinear Steiner tree problem when the allowed pathlength skew is infinite. Thus, heuristics for computing promising topologies are of interest.

[10] For the linear delay model, we use uniform loading capacitances in the input to the BB algorithm, because delay depends only on the edge lengths.

$x \le \max\limits_{p \in S}\{p.x\}$, $y + x \le \max\limits_{p \in S}\{p.y + p.x\}$. The *octagon set* of S, $Oct(S)$, is the set of sink locations in S that lie on the boundary of S's octagon.

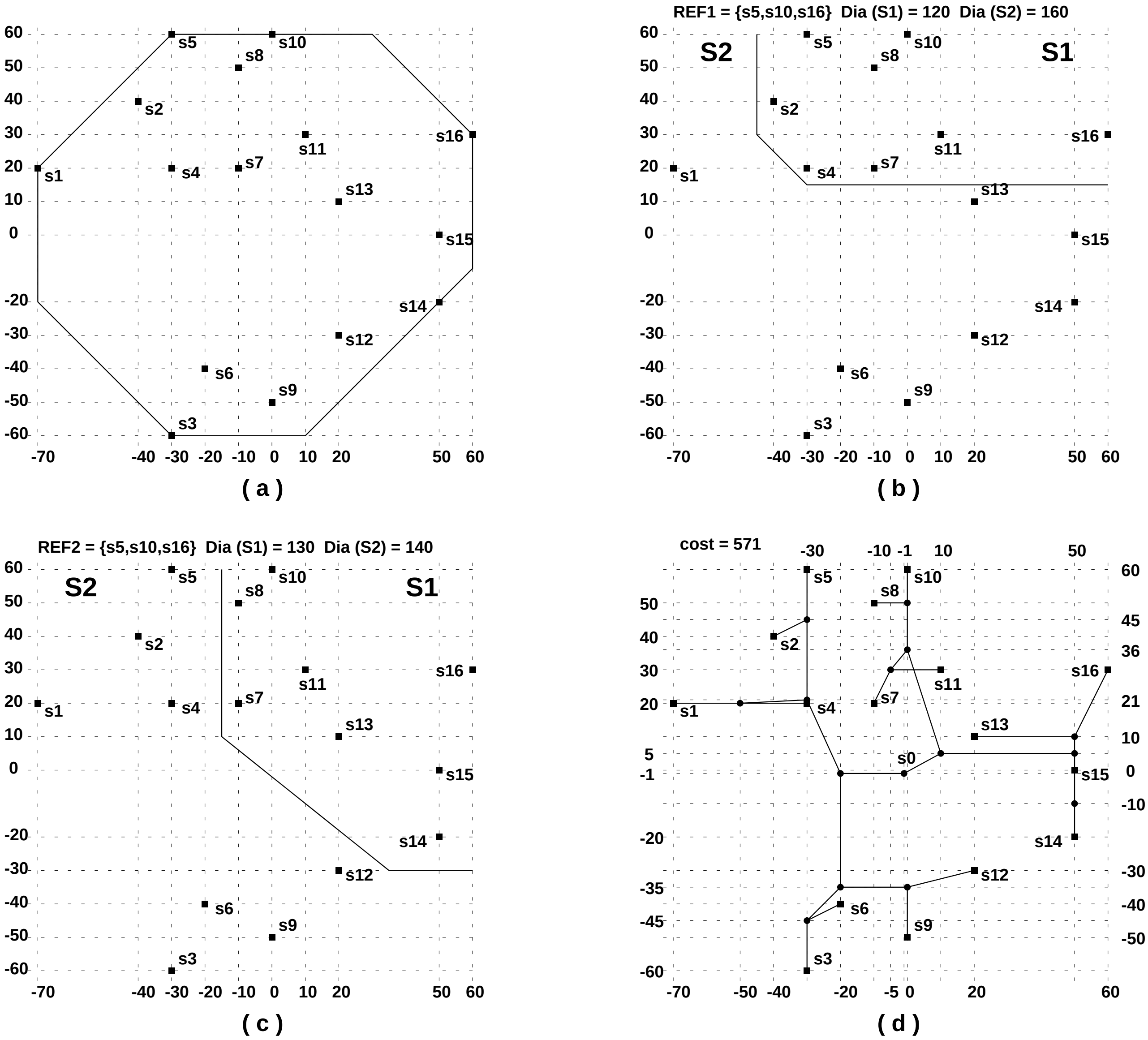

Figure 12: (a) Octagon lines of 16 sink locations; (b) Partition result of REF_1; (c) Partition result of REF_2; (d) The ZST produced by BB+DME.

Fig. 12(a) shows the octagon for a set of 16 sink locations; the octagon set is $\{s_1, s_3, s_5, s_{10}, s_{14}, s_{16}\}$. The lines defining the octagon induce a natural circular ordering on the sinks in the octagon set. For example, $s_1 - s_5 - s_{10} - s_{16} - s_{14} - s_3 - s_1$ is the circular order of the octagon set of Figure 12(a). Note that the octagon set construction naturally captures those parameters of the sink set which are relevant to diameter computations in the Manhattan plane. Based on extensive experimental investigations, we have found that each of the sets S_1 and S_2 in a balanced bipartition of S is likely to consist of consecutive elements in $Oct(S)$. Based on this observation, a balanced bipartition heuristic is as follows.

1. Compute $Oct(S)$ and sort $Oct(S)$ in circular order.

2. Perform steps 3-5 for each set of $\lfloor \frac{1}{2}|Oct(S)| \rfloor$ consecutive sinks in $Oct(S)$, called a *reference set* and denoted by REF_i, $i = 1, \ldots, |Oct(S)|$.

   3. For each sink $p \in S$, compute the *weight* of p, equal to $\min_{r \in REF_i} d(p, r) + \max_{r \in REF_i} d(p, r)$.

   4. Sort the sinks in ascending order of weight, then add sinks according to this order to S_1 until the difference between the sum of capacitances in S_1 and one half the total capacitance is minimized.

   5. The remaining sinks are placed in S_2, and the partition cost $dia(S_1) + dia(S_2)$ is obtained.

6. Over all reference sets REF_i, select the partition (S_1, S_2) with smallest partition cost.

In the example of Figure 12(a), each set of three consecutive sinks in the octagon set will be a possible reference set: $REF_1 = \{s_5, s_{10}, s_{16}\}$ has partition cost 280 as shown in Figure 12(b); $REF_2 = \{s_{10}, s_{16}, s_{14}\}$ has partition cost 270 as shown in Figure 12(c); etc. After all six reference sets have been evaluated, we find that the optimal reference set is REF_2 with cost 270. Figure 12(d) shows the output of the BB+DME algorithm on the instance of Figure 12(a).[11]

The time complexity of the BB algorithm is affected by characteristics of the sink set S. The number of times that the loop over steps 3 - 5 must be repeated is given by $|Oct(S)|$, the number of reference sets. In the worst case this value is $\Theta(n)$, but in practice it is usually bounded by a constant. Because BB is recursive, its complexity is also affected by the relative sizes of the bipartitions. In the worst case, when loading capacitances are very unbalanced, we can have $|S_1| = 1$ and $|S_2| = |S| - 1$.

Steps 3 and 4 dominate all others in the complexity of BB and are repeated for each reference set. (The diameters in step 5 can be calculated in linear time in the Manhattan metric.) Step 4 requires $O(n \log n)$ operations each time it is run, while step 3 requires $O(n|Oct(S)|)$ time. If $|Oct(S)| = \Theta(n)$, then the total time used in step 3 for a single bipartition can be reduced from $O(n^3)$ to $O(n^2 \log n)$ by using a priority queue such as a Fibonacci heap.[12]

In the very worst case, we can have $|Oct(S)| = \Theta(n)$ and pathologically unbalanced loading capacitances; each bipartition will require $O(n^2 \log n)$ time and the total time complexity of BB will be $O(n^3 \log n)$. If $|Oct(S)| = O(1)$ but loading capacitances are still unbalanced, the time complexity will be $O(n^2 \log n)$. The time complexity is reduced when we impose very reasonable constraints on the loading capacitances, e.g., the largest and smallest capacitances can differ by at most a constant factor, or simply that the cardinalities of the partitions differ by at most a constant factor. If the loading capacitances are "balanced"

[11] For the Elmore delay model, we observe that the DME algorithm is not always optimal for topologies generated by balanced bipartitioning. To see this, we modify the counter-example of Section 3.2 as follows. Let the loading capacitance of each sink be a small fixed value $\epsilon > 0$. Suppose that there are 16 sink nodes near point $s6$ within a very small radius $\delta > 0$ of each other. Similarly, suppose there are 8 sink nodes at point $s5$, 4 at $s4$, 2 at $s3$ and 1 at both $s1$ and $s2$. Then the BB algorithm will generate the topology of Figure 10.

[12] The priority queue, however, will increase the worst-case space requirements from $O(n)$ to $O(n^2)$.

and $|Oct(S)| = \Theta(n)$, then the time complexity of BB is $O(n^2 \log n)$. Finally, under the most realistic circumstances, when the loading capacitances are balanced and $|Oct(S)| = O(1)$, the time complexity of BB is $O(n \log^2 n)$.

4 Experimental Results

The BB and DME algorithms were implemented on Sun SPARC workstations in the C/UNIX environment. The code can be obtained from the authors. We compared routing cost and source-sink delay of the BB+DME output with previous results of Jackson et al. [17], Kahng et al. [18] and Tsay [25], which were obtained for both the linear and Elmore delay models.

Because the DME algorithm can be applied to any prescribed topology, we also applied it to topologies obtained in previous studies. In this way, we could separate the effects of DME from the effects of complementary heuristics for generation of clock tree topologies. We used two sets of benchmarks: (i) sink placements for the MCNC benchmarks Primary1 and Primary2 used in [17] and [18], and originally provided by the authors of [17] (Primary1 contains 269 sinks, and Primary2 contains 603 sinks); and (ii) sink placements for the five benchmark sets r1 - r5 used in [25] (the sizes of these examples range from 267 to 3,101 sinks).

	number of sinks	MMM cost	KCR cost	KCR+DME cost	reduction by KCR+DME from KCR (%)	BB+DME cost	reduction by BB+DME from MMM (%)	reduction by BB+DME from KCR (%)
Primary1	269	161.7	153.9	140.3	8.8	140.5	13.1	8.7
Primary2	603	406.3	376.7	350.4	7.0	360.8	11.2	4.2
r1	267	1,815	1,627	1,497	8.0	1,500	17.4	7.8
r2	598	3,625	3,349	3,013	10.0	3,010	17.0	10.1
r3	862	4,643	4,360	3,902	10.5	3,908	15.8	10.4
r4	1,903	9,376	8,580	7,782	9.3	8,000	14.7	6.8
r5	3,101	13,805	12,928	11,665	9.8	11,757	14.8	9.1
average					9.1		14.9	8.2

Table 2: Comparison of BB+DME with other algorithms in the linear delay model using MCNC benchmarks Primary1 and Primary2 and benchmarks r1 through r5 from Tsay.

4.1 Linear Delay Model

Our experimental results for linear delay are contained in Table 2. We compared BB+DME with the Method of Means and Medians (MMM) of Jackson et al. [17] and with the bottom-up, matching based method of Kahng, Cong and Robins (KCR) [18]. In order to test the performance of the DME algorithm alone, we also ran DME on the topologies produced by the KCR algorithm. The combined BB+DME algorithm produced an average reduction in cost of 15% from the MMM results. We also obtained an 8% average cost reduction from the KCR algorithm. Note that in the linear model, DME also produces trees with optimal source-sink *delay* [3], and our experiments showed an average reduction of 19% from the KCR algorithm. The improvement in source-sink delay ranged from 9% for Primary1 to 23% for r3.

	number of sinks	MMM cost	Tsay cost	Tsay+DME cost	KCR+DME cost	BB+DME cost	reduction by BB+DME from MMM (%)	reduction by BB+DME from Tsay (%)
Primary1	269	161.7	*	*	140.1	140.5	13.1	*
Primary2	603	406.3	*	*	345.2	360.8	11.1	*
r1	267	1,815	1,697	1,658	1,487	1,535	15.4	9.5
r2	598	3,625	3,432	3,368	3,020	3,065	15.4	10.7
r3	862	4,643	4,407	4,333	3,867	3,962	14.7	10.1
r4	1903	9,376	8,866	8,694	7,713	8,054	14.1	9.2
r5	3101	13,805	13,199	12,926	11,606	11,837	14.3	10.3
average							14.0	10.0

Table 3: Comparison of BB+DME with other algorithms in the Elmore delay model. *Results for Tsay's algorithm were obtained from Dr. Ren-Song Tsay and were not available for the Primary1 and Primary2 benchmarks.

4.2 Elmore Delay Model

We tested the BB+DME algorithm for Elmore delay on the same benchmark sink sets. The results are contained in Table 3. Again, these results indicat a significant improvement by BB+DME over previous algorithms. The average reduction in wirelength was 14% over MMM results, and 10% over the results of Tsay. It should be noted that DME alone resulted in an average improvement of only 2% over Tsay's algorithm, which can be attributed to the fact that Tsay's embedding algorithm allows deferral of the choice of placements for one level in the tree (the two endpoints of each merging segment are selected and carried to the next level, where the actual embedding is chosen to be the point which allows the minimum connection cost).[13] Our results also indicate a very significant reduction in source-sink delay in the Elmore model: the combination of KCR+DME reduced delay over the trees of Tsay by an average of 22%.

To obtain a more complete picture of the BB+DME performance, we also tested the algorithm on sink sets with locations chosen randomly from a square grid, i.e., with coordinates $s_i.x, s_i.y \in [-2500, 2500]$. The size of the sink sets ranged from 8 to 64. In these experiments, we also compared our algorithm with minimum rectilinear Steiner trees (RSTs) constructed by the heuristic in [7]; the BB+DME tree cost was only 64% above the heuristic RST cost. Finally, we used the circuit simulator SPICE2G.6 [20] to evaluate

[13] A surprising outcome of our experiments was the strong performance of topologies generated by the KCR algorithm. The combination of KCR and DME actually outperformed BB+DME by an average of 2.5% on the seven benchmarks. We expected balanced topologies to be superior in the Elmore delay model where the amount of load on each line affects delay, but our experimental results indicate that a bottom-up approach originally designed for the linear delay model can perform as well or better. However, we note that KCR uses such techniques as H-flipping and uncrossing of matching edges; the latter has exponential worst-case time complexity. Moreover, the minimum-diameter bipartitioning approach of BB is probably more useful when the distribution of sink locations is highly pathological.

clock skew in the ZSTs generated on the random sink sets. For both the MMM and BB+DME clock trees, SPICE decks were generated with the following specifications. The routing area was assumed to be $0.5cm \times 0.5cm$, and all the parameters were based on a $1.2\mu m$ CMOS technology. An input clock frequency of 100 MHz and a superbuffer driven by the input clock source were assumed. The delays between the source and the sink nodes were measured at the output node of the inverter which drives the sink nodes. Table 4 shows the average maximum delays, minimum delays and clock skews for the sinks sets of each size. The maximum delay of BB+DME was on average 3% less than that of MMM. The average skew of MMM was 9.2 picoseconds while that of BB+DME was only 0.5 picoseconds, a 93% reduction. Figure 13 shows the output of the BB+DME algorithm on an instance containing 64 sinks. The total routing length is $50445\mu m$ and the source-sink delay is 0.91ns. By contrast, the MMM algorithm yielded a tree with cost $59256\mu m$ and delay 0.94ns for this case.

#Pts	MMM			BB+DME			BB+DME / MMM		
	delay		clock	delay		clock	delay		clock
	max	min	skew	max	min	skew	max	min	skew
8	769.3	763.2	6.1	746.6	746.2	0.4	0.970	0.978	.07
16	801.8	797.0	4.8	783.2	782.5	0.7	0.977	0.982	.15
24	836.6	826.2	10.4	808.7	808.3	0.4	0.967	0.978	.04
32	863.5	855.6	7.9	837.3	836.5	0.8	0.970	0.978	.10
40	885.6	876.3	9.3	857.0	856.5	0.5	0.968	0.977	.05
48	908.9	896.4	12.5	876.8	876.3	0.5	0.965	0.978	.04
56	926.2	914.4	11.8	890.2	889.7	0.5	0.961	0.973	.04
64	940.6	930.1	10.5	910.7	910.2	0.5	0.968	0.979	.05
average							0.968	0.978	.07

Table 4: Mean delay time and clock skew for random sink sets (time unit = picosecond). The rightmost three columns display ratios between the results of BB+DME and MMM.

5 Conclusions and Directions for Future Work

Minimization of clock skew is critical to the design of high-performance VLSI systems. Recent research has yielded a number of heuristics which effectively eliminate skew according to either the Elmore or linear delay model. However, these previous methods concentrate on generation of the clock tree topology, and then embed the topology in the plane with little concern for the minimization of total wirelength.

Obviously, minimization of total wirelength will lead to reduction of wiring area, with the added effect of less blockage for subsequent routing phases of layout. We also note that clocking accounts for a large portion of system power requirements: wire minimization can significantly reduce the power needed to drive the clock signal, thus improving system feasibility and reliability. Finally, wirelength reduction will improve performance by lessening such effects as pulse narrowing, pulse deformation, etc. Given these considerations, our work gives a unified approach to clock tree construction which combines the topology

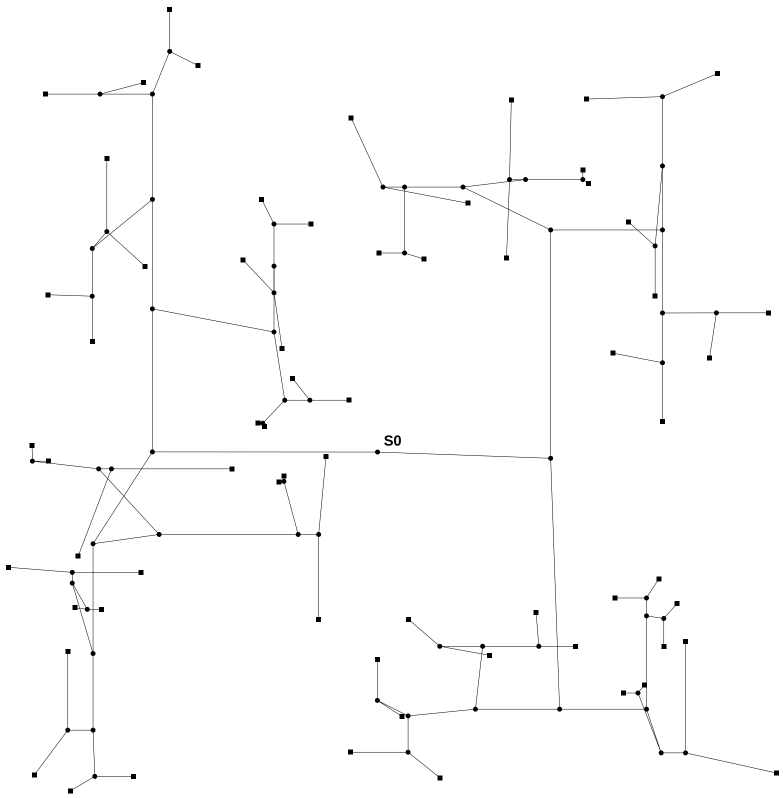

Figure 13: An example of a ZST produced by BB+DME for 64 randomly chosen sink nodes.

generating phase (BB) with the embedding phase (DME).

The balanced bipartition (BB) heuristic generates a connection topology by recursively dividing the set of sinks into two subsets with equal total loading capacitance while at the same time minimizing the sum of diameters of the two subsets. This balance condition is a novel aspect of the method, and is useful when delay depends on both pathlength and capacitance, as in the Elmore model. The partitioning strategy based on minimizing the sum of diameters improves upon previous top-down bisection strategies of Jackson et al. [17] and Tsay [25], which can only use horizontal or vertical cuts to partition the set of sinks.

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 alone to the clock trees constructed by other algorithms results in wirelength reductions of 2% to 9%. The DME algorithm also extends to problem formulations where the clock source is prescribed. Finally, given the linear delay model, DME yields *optimal* total wirelength and optimal source-sink delay.

Our experimental results indicate that the BB+DME methodology yields routing solutions with exact zero skew (which we confirmed to be in the subpicosecond range using SPICE2G.6) and significantly reduced total wirelengths (8% - 15% less than the best previous methods). Furthermore, the superiority of BB+DME over previous methods depends on their joint application. For instance, our improvement of approximately 8% over the matching-based method of Kahng et al. (KCR) [18] is directly attributable to the DME embedding, since DME applied to topologies generated by KCR yields clock tree cost very similar to that obtained using BB+DME. On the other hand, DME alone can achieve only 2% out of the 15% improvement of BB+DME over Tsay [25]. Thus 13% of the cost savings can be attributed to the BB topology.

There are many promising extensions to our current approach. The DME algorithm readily applies to problems of *prescribed* skew (i.e., "useful" skew [1]), where the arrival times of the clocking signal must differ by prescribed amounts. This is handled by setting initial delays at the sinks to non-zero values. The DME algorithm can also be used for problems with *allowed* skew [1] [13] [25], where the signal must arrive at each sink within a prescribed segment of time.

Finally, the general issue of topology generation remains an important area for further investigation. A promising approach is to run DME concurrently with matching-based and other bottom-up topology generating heuristics. In general, the construction of optimal topologies appears to be very difficult (perhaps NP-hard). However, we expect future investigations in this area to have fruitful applications, for both clock tree construction and the broader area of high-performance routing.

6 Remarks and Acknowledgements

Through independent research, the two groups of authors came up with essentially identical approaches to constructing zero skew clock routing trees with minimum wirelength for a given tree topology. The major differences between the two treatments are: (i) Chao, Hsu and Ho apply DME to the Elmore delay model, while Boese and Kahng establish the theoretical results for DME with respect to both the linear and Elmore delay models; and (ii) Chao, Hsu and Ho proposed the top-down balanced bipartition technique to generate an initial clock tree topology, while Boese and Kahng assume arbitrary existing tree topologies, e.g., those derived from the KCR method [18] [9]. The work of Chao, Hsu and Ho [8] appeared at the 29th ACM/IEEE Design Automation Conference; the work of Boese and Kahng [3] appeared at the 5th IEEE International Conference on ASIC. The authors are grateful to Dr. Ren-Song Tsay for providing benchmark data and for his communications which made this collaboration possible.

7 Appendix: Proofs of Lemmas 1, 2, and 3

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 the TRR $R_1 \cap R_2$ is a Manhattan arc.*

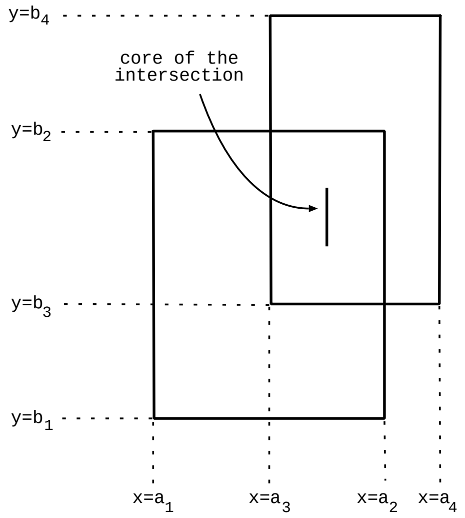

Figure 14: Intersection of two TRRs after 45-degree rotation.

Proof: Rotate the plane by 45 degrees so that the boundaries of R_1 and R_2 are vertical and horizontal line segments (see Figure 14). Let R_1' and R_2' be the two TRRs after rotation with boundary lines given by:

- 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

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

Since rotating each TRR by 45 degrees requires constant time, determining the intersection of the two

TRRs $R_1 \cap R_2$ also requires only 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 become 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 line segment or a single point. Since $R_1 \cap R_2$ is also a TRR, it must be a Manhattan arc. $\square$

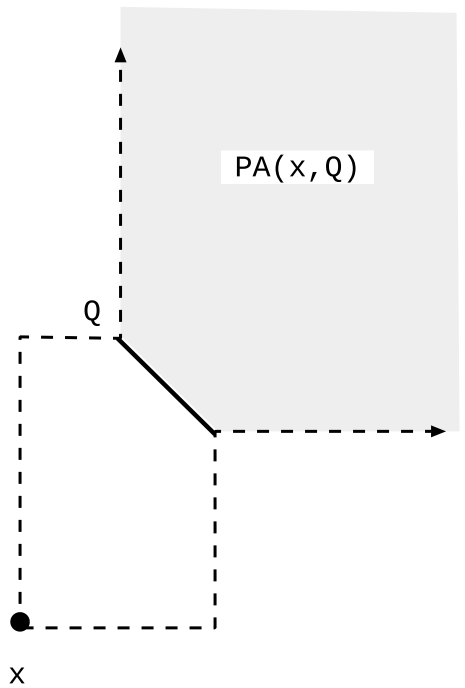

Figure 15: Projection area $PA(x, Q)$ under the Manhattan metric.

Define a *straight-line path* between two points x and y to be any minimum-length path between them using only vertical and horizontal lines. 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 them. 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 15 contains an example of the projection area from a point x through a Manhattan arc Q.

The next lemma about projection areas will be used to prove Lemmas 2 and 3. It states that the union of two projection areas from points p and q, respectively, through a merging segment ms between them, is the entire plane.

Lemma 4 : *Let ms be a merging segment between the two points p and q. Then*

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

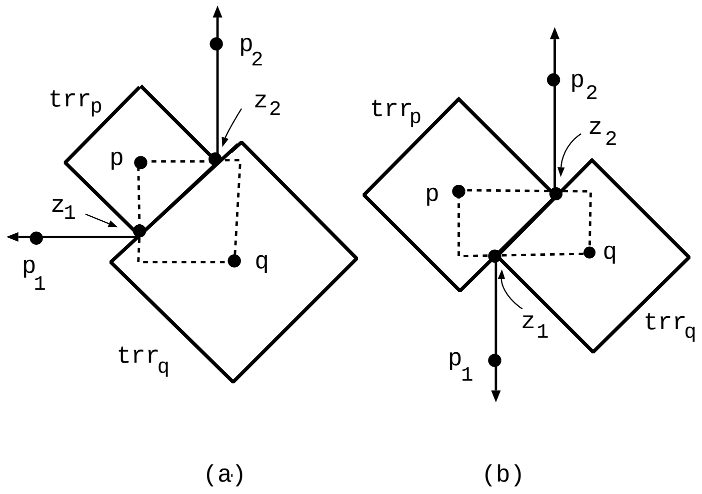

Figure 16: Two cases to consider in the proof of Lemma 4.

Proof: If the merging cost between p and q is greater than $d(p,q)$, then either $ms = \{p\}$ or $ms = \{q\}$. Since for any point x, $PA(x, \{x\}) = \Re^2$, this implies that either $PA(p, ms) = \Re^2$ or $PA(q, ms) = \Re^2$ and the proof is complete. For the case when the merging cost equals $d(p,q)$, 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

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

for some x satisfying $0 \leq x \leq 1$. If $x = 1$ or $x = 0$, the lemma is immediately true, since either $PA(p, \{p\}) = \Re^2$ or $PA(q, \{q\}) = \Re^2$ will hold. Let z_1 and z_2 be the two endpoints of merging segment ms. If $0 < x < 1$ then we need to consider the two cases depicted in Figure 16:

(a) z_1 and z_2 are both corners of the same TRR, either trr_p or trr_q. Assume without loss of generality that they are both corners of trr_p.

(b) z_1 and z_2 are corners of different TRRs. Assume without loss of generality that z_1 is a corner of trr_q and z_2 is a corner of trr_p.

Define a *ray* $\overrightarrow{p_1p_2}$ from point p_1 through point p_2 as the half-line with endpoint p_1 that extends through p_2. In case (a), the straight-line path from p to z_1 is a vertical line segment and the straight-line path from p to z_2 is a horizontal segment. In Figure 16(a) it is evident that $PA(p, \{z_1\})$ is a half plane with border line z_1p_1 and $PA(p, \{z_2\})$ is a half plane bordered by line z_2p_2. Furthermore, $PA(p, ms)$ is the infinite region separated from p by (and including) ray $\overrightarrow{z_1p_1}$, segment ms, and ray $\overrightarrow{z_2p_2}$. Similarly, $PA(q, ms)$ is the region separated from q by the same border. Consequently, $PA(p, ms) \cup PA(q, ms)$ is the entire plane.

In case (b), shown in Figure 16(b), $PA(p, ms)$ is the infinite region separated from p by (and including) $\overrightarrow{z_1p_1}$, ms, and $\overrightarrow{z_2p_2}$. $PA(q, ms)$ is the region separated from q by the same border. Again, $PA(p, ms) \cup$

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

Lemma 2 : *Given a ZST T with topology G, let v be an internal node with children a and b. Suppose the subtrees of T rooted at a and b can be generated by the DME algorithm for some placement of v on $ms(v)$, and also suppose that $q = pl(T, v) \notin ms(v)$. Then a new ZST T' with the same topology can be constructed from T by moving the placement of v so that the following hold: (i) $q' = pl(T', v) \in ms(v)$; (ii) $cost(T') < cost(T)$; and (iii) $t_{LD}(T, q) = t_{LD}(T', q)$.*

Proof: Consider Figure 8 of Section 3.2.2. Let a and b be the placements in T of v's children. By Lemma 4, there exists a point q' on $ms(v)$ such that there is a straight-line path either 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. Because $bq'q$ is a straight-line path, segment bq in T can be replaced by segments bq' and $q'q$ in T' without changing the delay between b and q, and leaving the delay at point q unchanged. Moreover, the construction of $ms(v)$ ensures that zero-skew is maintained by setting the edge e_a equal to the segment aq' and $pl(T', v) = q'$. Define $length(T, xy)$ to be the edge length between points x and y in ZST T. Because the delay at q remains unchanged in T' and the a—q and b—q connections share wire between q' and q in T', we must have $cost(T') = cost(T) - length(T', q'q)$. $\square$

Lemma 3 : *Suppose that a and b are two sibling nodes in ZST T with parent v, and suppose 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 ZST T' can be constructed from the same topology, with $cost(T') < cost(T)$ and with $t_{LD}(T, q) = t_{LD}(T', q)$ for $q = pl(T, v)$.*

Proof: (See Figure 9 in Section 3.2.2.) To prove the lemma, we will first construct a ZST T_{new} with source at $q = pl(T, v)$, and then replace the subtree of T rooted at v with part of T_{new} to create T'. Using Theorem 2 of [3] we show that the connections a—q and b—q share wire on a partial edge $e_{q'}$ in T', whereas they do not share wire in T. Because T' is also constructed so that the lengths of the a—q and b—q connections are the same as in T, tree T' will have lower cost than T.

Let G_v be the subtree of topology G rooted at v, and let S_v be the set of sinks in G_v. Suppose that sink s_i is the sink in S_v furthest from q. Create a new sink z that is located at a point directly opposite of q from s_i; i.e., $d(q, s_i) = d(q, z)$ and $d(s_i, z) = 2 * d(q, s_i)$. Consider a new set of sinks: $S_{new} = S_v \cup \{z\}$.

We create a topology G_{new} for S_{new} that merges G_v and z at its root, s_{new0}. We then run DME on S_{new} using topology G_{new} to create ZST T_{new}. By Theorem 2 of [3], T_{new} will have minimum feasible delay at each sink, equal to one-half the diameter of S_{new}, specifically $d(q, s_i)$. By the Fact used in the proof of Theorem 2 of [3], $ms(s_{new0})$ is the set of all points within distance $d(q, s_i)$ of every sink in S_{new}. Therefore, $q \in ms(s_{new0})$ and T_{new} can be constructed so that $q = pl(T_{new}, s_{new0})$. Let $a' = pl(T_{new}, a)$,

$b' = pl(T_{new}, b)$, and $q' = pl(T_{new}, v)$. We now construct ZST T' for S by cutting off the subtree of T rooted at q and replacing it with T_{new} minus the edge between q and z. Since $t_{LD}(T', q) = d(q, s_i)$, it must be that $t_{LD}(T', q) \leq t_{LD}(T, q)$. If the strict inequality holds, we add extra wire between q and q' to enforce equality, and thereby retain zero skew.

For convenience, we use $e_{a'}$ and $e_{b'}$ to represent the embeddings of edges e_a and e_b in T'. We also use $e_{q'}$ to denote the partial edge between q' and q in T'. Because the subtrees of T rooted at a and b were constructed according to DME, we have $t_{LD}(T, a) = t_{LD}(T', a')$ and $t_{LD}(T, b) = t_{LD}(T, b')$. Thus, because $t_{LD}(T', q) = t_{LD}(T, q)$, it must be that

$$|e_a| = |e_{a'}| + |e_{q'}| \quad \text{and} \quad |e_b| = |e_{b'}| + |e_{q'}|. \tag{3}$$

Because $d(a, b) > d(ms(a), ms(b))$ and $d(a, b) > |t_{LD}(a) - t_{LD}(b)|$, $d(a, b)$ is strictly greater than the merging cost between $ms(a)$ and $ms(b)$. Therefore,

$$|e_a| > |e_{a'}| \quad \text{and} \quad |e_b| > |e_{b'}|. \tag{4}$$

Equations (3) and (4) imply that $|e_{q'}| > 0$, and thus

$$|e_a| + |e_b| \quad > \quad |e_{a'}| + |e_{b'}| + |e_{q'}|.$$

As a result, $cost(T') < cost(T)$. $\square$

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*, 1986, pp. 118-122.

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

[3] K. D. Boese and A. B. Kahng, “Zero-Skew Clock Routing Trees With Minimum Wirelength,” *Proc. IEEE Intl. Conf. on ASIC*, 1992, pp. 1.1.1 - 1.1.5.

[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*, 1989, pp. 15.4.1-15.4.4.

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

[6] P. K. Chan and K. Karplus, “Computing Signal Delay in General RC Networks by Tree/Link Partitioning”, *IEEE Trans. on CAD* 9(8), August 1990, pp. 898-902.

[7] T.-H. Chao and Y.-C. Hsu, “Rectilinear Steiner Tree Construction by Local and Global Refinement,” *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 432-435.

[8] T.-H. Chao, Y.-C. Hsu and J.-M. Ho, “Zero Skew Clock Net Routing,” in *Proc. ACM/IEEE Design Automation Conf.*, 1992, pp. 518-523.

[9] J. Cong, A. B. Kahng and G. Robins, "Matching-Based Methods for High-Performance Clock Routing", to appear in *IEEE Transactions on CAD*.

[10] 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.

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

[12] W. C. Elmore, "The Transient Response of Damped Linear Networks With Particular Regard to Wide-Band Amplifiers," *Journal of Applied Physics* 19(1), Jan. 1948, pp. 55-63.

[13] J. P. Fishburn, "Clock Skew Optimization," *IEEE Transactions on Computers* 39(7), July 1990, pp. 945-951.

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

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

[16] A. Hanafusa, Y. Yamashita and M. Yasuda, "Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards," *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 386-389.

[17] 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.

[18] 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.

[19] C. Monma and S. Suri, "Partitioning Points and Graphs to Minimize the Maximum or the Sum of Diameters," *Proc. Sixth Intl. Conf. on the Theory and Applications of Graphs*, John Wiley & Sons, 1988.

[20] L. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," *ERL Memo. No. UCB/ERL M75/520*, May 1975.

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

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

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

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

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

[26] 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.