Zero Skew Clock Routing with Minimum Wirelength

Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, Kenneth D. Boese, Student Member, IEEE, and Andrew B. Kahng, Associate Member, IEEE

Abstract—In the design of high performance VLSI systems, minimization of clock skew is an increasingly important objective. Additionally, wirelength of clock routing trees should be minimized in order to reduce system power requirements and deformation of the clock pulse at the synchronizing elements of the system. In this paper, we present the deferred-merge embedding (DME) algorithm, which 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] and [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.

I. 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]:

$$\text{clock period} \geq t_d + t_{\text{skew}} + t_{\text{in}} + t_{\text{sl}}$$

where $t_d$ is the delay on the longest path through combinational logic, $t_{\text{skew}}$ is the clock skew, $t_{\text{in}}$ is the setup time of the synchronizing elements (assuming edge triggering), and $t_{\text{sl}}$ is the propagation delay within the synchronizing elements. The term $t_d$ can be further decomposed into $t_d = t_{d,\text{interconnect}} + t_{d,\text{gates}}$, where $t_{d,\text{interconnect}}$ is the delay associated with the interconnect of the longest path through combinational logic, and $t_{d,\text{gates}}$ is the delay through the combinational logic gates on this path. Increased switching speeds due to advances in VLSI fabrication technology will significantly decrease the terms $t_{\text{in}}$, $t_{\text{sl}}$, and $t_{d,\text{gates}}$. Therefore, $t_{d,\text{interconnect}}$ and $t_{\text{skew}}$ become the dominant factors in determining circuit performance: Bakoglu [2] has noted that $t_{\text{skew}}$ 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, Ramanathan and Shin [21] have proposed a clock distribution scheme that 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. [9], [18] 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 [9] and [18] 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 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 [9], [18], [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 that 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 IV 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]. 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 routing density must be minimized.

The remainder of this paper is organized as follows. In Section II, 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 III presents our main results. These include: i) the 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 BB strategy to derive a good tree topology to which the DME algorithm may be applied. Section IV gives experimental results and comparisons with previous work, and Section V concludes with directions for future research.

II. 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 \( S = \{ s_1, s_2, \ldots, s_j \} \subset \mathbb{R}^2 \), specifies an

\[ \text{Note that SPICE simulations for BB + DME constructions on random sink sets (Table IV below) indicate only a 3% improvement in delay compared to the MMM algorithm. This suggests that although the Elmore model is reasonably accurate for predicting skew, it is less accurate for predicting delay.} \]
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 \( \mathbb{R}^2 \) with each node \( v \in G \); we will use \( p(T, v) \) or \( p(v) \) to represent this location. (When no confusion arises, we may also denote \( p(T, v) \) 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 \( e_{vw} \). 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 \( \text{children}(v) \). The wirelength, or cost, of the edge \( e_{vw} \) is denoted by \( |e_{vw}| \) and must be greater than or equal to the Manhattan distance between its endpoints \( p(w) \) and \( p(v) \). The cost of \( T(S) \), denoted \( \text{cost}(T(S)) \), is the total wirelength of the edges in \( T(S) \).

For a given clock tree \( T(S) \), let \( t_f(s_0, s_j) \) denote the signal propagation time, or delay, on the unique path from source \( s_0 \) to sink \( s_j \); the collection of edges in this path is denoted by \( \text{path}(s_0, s_j) \). The skew of \( T(S) \) is the maximum value of \( t_f(s_0, s_j) - t_f(s_0, s_i) \) over all sink pairs \( s_j, s_i \in S \). If \( T(S) \) has zero skew 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.

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 linear model or the Elmore model, both of which are easier to compute and optimize during clock tree design.

\(^2\) 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 \( G \).

\(^4\) To route a wire of length greater than the distance between its endpoints, the method of specified-length routing due to Hanafusa et al.,\(^{16}\) can be used.

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. Delay Models

2.1.1. Linear Delay: In the linear delay model, the delay along \( \text{path}(s_0, s_j) \) 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

\[
t_{LD}(u, w) = \sum_{e_{uv} \in \text{path}(u, w)} |e_{uv}|
\]

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 design 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 \( \alpha \) and \( \beta \), respectively, denote the resistance and capacitance per unit length of interconnect, so that the resistance \( r_e \) and capacitance \( c_e \) of edge \( e_{vw} \) are given by \( \alpha \cdot |e_{vw}| \) and \( \beta \cdot |e_{vw}| \), respectively. For each sink \( s_j \) in the tree \( T(S) \), there is a loading capacitance \( C_L \) which is the input capacitance of the functional unit driven by \( s_j \).

We let \( T_v \) denote the subtree of \( T(S) \) rooted at \( v \), and let \( c_v \) denote the node capacitance of \( v \). The 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:

\[
C_v = \begin{cases} 
  c_v, & \text{if } v \text{ is a sink node } s_j \\
  c_v + \sum_{e \in \text{children}(v)} (c_e + C_e), & \text{if } v \text{ is an internal node}.
\end{cases}
\]

According to\(^{12, 22, 23}\), the Elmore delay \( t_{EL}(s_0, s_j) \) can be calculated by the following formula (see\(^{25}\) for a discussion of underlying circuit models):

\[
t_{EL}(s_0, s_j) = \sum_{e_{uv} \in \text{path}(s_0, s_j)} r_e (\frac{1}{2} c_e + C_v).
\]

More generally, the delay time between any two vertices \( u \) and \( w \) is

\[^3\] As noted earlier, we will assume that \( c_e = 0 \) for each internal node in all of our examples and benchmarks.
and $w$ on a source-sink path is given by

$$t_{ED}(u, w) = \sum_{e \in \text{path}(u, w)} r_e \left( \frac{3}{2} c_e + C_e \right).$$

Elmore delay is additive: if $v$ is a vertex on the $u \rightarrow w$ 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 $u \rightarrow s_j$ path, then $t_{ED}(u, s_j) = r_e \left( \frac{1}{2} c_e + C_e \right) + t_{ED}(v, s_j)$. A sink node $s_j$ may be treated as a trivial zero skew subtree with capacitance $c_{s_j}$ and delay zero.

III. 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 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 BB algorithm that creates a good connection topology, leading to the unified BB + DME algorithm.

3.1. The DME Algorithm

The 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 $T_a$ and $T_b$ to denote the subtrees of merging segments rooted at $a$ and $b$, respectively. We are interested in placements of $v$ which allow $T_a$ and $T_b$ to be merged with minimum added wire while preserving zero skew. Define the merging cost between $T_a$ and $T_b$ 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 $p(v)$. Because delay is a monotone increasing function of wire-length, there is a unique optimal assignment of lengths to $e_a$ and $e_b$.

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 $45^\circ$ from the wiring directions. The collection of points within a fixed distance of a Manhattan arc is called a tilted rectangular region, or TR, whose boundary is composed of Manhattan arcs (see Fig. 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.

The merging segment of node $v$, $ms(v)$, is defined recursively as follows: if $v$ is a sink $s_j$, then $ms(v) = \{s_j\}$. If $v$ is an internal node, then $ms(v)$ is the set of all placements $p(v)$ which allow minimum merging cost, that is to say, all points that are both within distance $|e_v|$ of $ms(a)$ and within distance $|e_v|$ 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, $\tau_{r_a}$ with core $ms(a)$ and radius $|e_v|$, and $\tau_{r_b}$ with core $ms(b)$ and radius $|e_v|$; i.e., $ms(v) = \tau_{r_a} \cap \tau_{r_b}$.

The merging cost at $v$ has an obvious lower bound of $\kappa - d(ms(a), ms(b))$. If the merging cost is greater than $\kappa$ (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. Fig. 2 illustrates the algorithm for the case where the merging cost is equal to $\kappa$, and Fig. 3 illustrates the algorithm for the case where the merging cost is greater than $\kappa$. An entire tree of merging segments is illustrated by Fig. 4. The leaves of the tree of segments are all single points representing the sink locations $s_1, \ldots, s_n$, 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.)

**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 $(R_1) + radius (R_2) = d(\text{core}(R_1), \text{core}(R_2))$, 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 $\kappa$, then $d(\text{core}(\tau_{r_a}), \text{core}(\tau_{r_b})) = |e_v| + |e_v| = radius(\tau_{r_a}) + radius(\tau_{r_b})$, and hence, $\tau_{r_a} \cap \tau_{r_b}$ is a Manhattan arc; or ii) if the merging cost at $v$ is greater than $\kappa$, then either $\tau_{r_a}$ or $\tau_{r_b}$ will be a Manhattan arc whose intersection with any convex set will also be a Manhattan arc. For each sink $s_j$, the merging segment $ms(s_j)$ is a single point and

6The uniqueness is shown as follows. Suppose the minimum merging cost is $c$. Define a function $f(|e_v|)$ to be the path delay from $v$ to sinks in $T_v$ for edge length $|e_v|$, similarly define $g(|e_v|)$ for the path delay from $v$ to sinks in $T_w$. Define $g(|e_v|) = g(|e_v|) - c - f(|e_v|)$. A length assignment to $e_v$ must satisfy $f(|e_v|) - g(|e_v|) = 0$, or alternatively, $(f - g)|e_v| = 0$. If both $f$ and $g$ are monotone increasing functions, then $g$ is monotone decreasing and $f - g$ is monotone increasing. Thus $(f - g)|e_v| = 0$ will have at most one solution.
thus a Manhattan arc. By induction, therefore, all merging segments must be Manhattan arcs.

Fig. 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.

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 $pl(v)$; 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 Fig. 6). For the tree of merging segments in Fig. 4, the resulting placements are indicated by the points at which the segments are connected by dotted lines. Fig. 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
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 ≠ n_o

For each node v in G (bottom-up order)
if v is a sink node,
ms(v) ← pl(v)
else
Let a and b be the children of v
Calculate Edge Lengths of (a, b)
Create TRUs τ_{a,b} and τ_{b,a} as follows:
core(τ_{a,b}) ← ms(a)
radius(τ_{a,b}) ← |e_a|
core(τ_{b,a}) ← ms(b)
radius(τ_{b,a}) ← |e_b|
ms(v) ← τ_{a,b} ∪ τ_{b,a}
endif

Fig. 5. Construction of the tree of segments.

Fig. 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: STS(T)
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
Connect τ_{p,v} as follows:
core(τ_{p,v}) ← pl(p)
radius(τ_{p,v}) ← |e_v|
Choose any pl(v) ∈ ms(v) ∩ τ_{p,v}
endif

Fig. 7. Construction of the ZST by embedding internal nodes of the
topology.

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_{L,D}(a) and t_{L,D}(b) be
the delays from a and b to the sinks in their respective
subtrees. Then, zero skew at v requires that

\[ t_{L,D}(a) + |e_a| = t_{L,D}(b) + |e_b| \]

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

\[ |e_a| = \frac{\kappa + t_{L,D}(b) - t_{L,D}(a)}{2} \]

and

\[ |e_b| = \kappa - |e_a| \]

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

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 ZST’s 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 corre-
spond 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 \( t_{L,D}(T, x) \) denote
the delay in ZST T between a point x in T and each sink
which has x on its source-sink path.

**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)
\( t_{L,D}(T, q) = t_{L,D}(T', q) \).

Lemma 2 is illustrated in Fig. 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 \( d(a, b) >
|t_{L,D}(T, a) - t_{L,D}(T, b)| \), then a new ZST T’ can be
constructed from the same topology, with cost(T’) < cost(T)
and with \( t_{L,D}(T, q) = t_{L,D}(T', q) \) for \( q = pl(T, v) \).

Fig. 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) $p(v) \in ms(v)$ and ii) $d(p(v), p(p)) \leq L_v$, 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 $p(T, v)$ must be in $ms(v)$ and $p(T, w)$ must be in $ms(w)$. Thus $v$ does not violate condition i).

Consequently, $v$ must violate condition ii), i.e., $d(p(T, v), p(T, w)) > L_v$. 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 $d(p(T, v), p(T, w)) \leq d(ms(v), ms(w))$. 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 $d(p(T, v), p(T, w)) \leq |l_{LD}(v) - l_{LD}(w)|$, then cost ($T$) can be reduced by moving $p$ to $p(v)$ if $L_v = 0$ or to $p(w)$ if $L_w = 0$. Thus, we have $d(p(T, v), p(T, w)) > d(ms(v), ms(w))$, and $d(p(T, v), p(T, w)) > |l_{LD}(v) - l_{LD}(w)|$. Then by Lemma 3 cost ($T$) can be decreased, contradicting the assumption that $t$ has minimum cost.

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 predefined. 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 $p(s_0)$. 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

We use the analysis of Tsay [25] 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. Let $TS_a$ and $TS_b$, respectively, have capacitance $C_1$ and $C_2$, and delay $t_1 = t_{ED}(a)$ and $t_2 = t_{ED}(b)$; let $p(v)$ be a merging point with minimum merging cost. From the definition of Elmore delay, we have that $t_{ED}(w, a) = r_{e}(1/2)c_{e_a} + C_1$. Thus, $p(v)$ satisfies

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

Let $d(ms(a), ms(b)) = \kappa$. Suppose that $TS_a$ and $TS_b$ can be merged with merging cost $\kappa$; in other words, $|e_a| = x$ and $|e_b| = \kappa - x$ for $0 \leq x \leq \kappa$. 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

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

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

Case 2: If $x < 0$ or $x > \kappa$, then the assumption of merging cost $\kappa$ results in a negative edge length for either $e_{s_2}$ or $e_{s_4}$. 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 $p(a)$ as the merging point and set $|e_{s_1}| = 0$ and $|e_{s_2}| = \kappa'$. Then

$$t_1 = \alpha \kappa' \left( \frac{1}{2} \kappa' + C_z \right) + t_2$$

and we use the quadratic formula to solve for $\kappa'$:

$$\kappa' = \frac{\left( a C_z^2 + 2 \alpha \beta (t_1 - t_2) \right)^{1/2} - a C_z}{a \beta}.$$  

Similarly, if $x > \kappa$, we set $|e_{s_4}| = 0$ and

$$|e_{s_2}| = \kappa' = \frac{\left( a C_1^2 + 2 \alpha \beta (t_2 - t_1) \right)^{1/2} - a C_1}{a \beta}.$$ 

The above analysis shows that a zero skew merging point between two ZST's can always be found. The merging cost depends on the distance between the roots of the ZST's, 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 partition 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 model.

3.3.2. Suboptimality of DME for Elmore Delay: 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 IV 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 ZST's $T$ in Fig. 10 and $T'$ in Fig. 11 demonstrate that, for some sink sets and topologies, DME will not be optimal for Elmore delay. $T$ and $T'$ connect terminal points $s_1, \ldots, s_5$ to source $s_0$. Both trees are assumed to extend to the right side of $s_0$, with their subtrees to 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 $\alpha$ and unit capacitance $\beta$ to one, and the loading capacitance $C_{s_0}$ of each sink node $s$ to zero.$^3$

The ZST $T'$ in Fig. 11 is constructed so that if points $s_1$ and $s_2$ are merged at point $p_1$, then vertical wires from points $s_1$ through $s_2$ will merge along the horizontal wire from $s_1$ to $s_2$ with exactly zero skew. If, however, $s_1$ and $s_2$ are merged on their merging segment as shown in the tree $T$ of Fig. 10, the delay at $p_1$ will increase, and jogs will be required in the edges $e_{s_1}$ through $e_{s_4}$. In this example, the four required jogs are each of length greater than 0.3. Thus, their sum is greater than 0.1, which was the amount of wire saved initially by merging $s_1$ and $s_2$ at $p_0$.

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

Fig. 11. ZST $T'$, which has optimal cost for the topology in Fig. 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 I 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

$$t_{ED}(p_2') = t_{ED}(p_1') + 0.1 \left( 0.1 + C_{p_1} \right) = 60.5 + 3.305 = 63.8.$$
Because unit resistance and capacitance both equal one, and because loading capacitances at the leaves are zero, the tree capacitance of a node is given by the amount of wire in its subtree. Thus, we see in Table I that cost ($T'$) is less than cost ($T$) by 0.44.

### 3.4. Topology Generation

It is easy to see that, as hinted by the examples of Figs. 10 and 11, the choice of topology will affect the performance of the DME embedding. We now present a new heuristic for generating connection topologies.\(^5\) 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 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 \(\text{dia}(S)\), the number of sinks in $S$ by $|S|$, and the number of sinks in $S$ by $\Sigma$. Since the cost of any routing tree of $S$ is greater than $\text{dia}(S)$ and less than $2 \times \text{dia}(S)$, we consider $\text{dia}(S)$ 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 partition of a set of sinks $S$ into two subsets $S_1$ and $S_2$ a balanced partition if the difference between the total loading capacitances of the two subsets is at most $\text{dia}(S_1)$.\(^6\) Intuitively, we would like to find a balanced partition which divides set $S$ with minimum partition cost, given by $\text{dia}(S_1) + \text{dia}(S_2)$. This is the idea behind the BB heuristic. In the Euclidean metric, the problem of constructing a balanced partition which minimizes the sum of diameters can be solved in $O(n^3)$ time [19]. However, we are not aware of any polynomial-time algorithm that yields a minimum cost balanced partition 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 clockwise order around the octagon): $y \leq \min_{p \in S} \{p, y\}$, $y \leq \min_{x \in S} \{p, x\}$, $x \leq \min_{x \in S} \{p, x\}$, $y \leq \max_{p \in S} \{p, y\}$, $y \leq \max_{p \in S} \{p, y\}$, $x \leq \max_{x \in S} \{p, x\}$, $y \leq \max_{p \in S} \{p, y\}$. The octagon set of $S$, Oct ($S$), is the set of sink locations in $S$ that lie on the boundary of $S$'s octagon.

Fig. 12(a) shows the octagon for a set of 16 sink locations; the octagon set is $\{s_1, s_2, s_3, s_4, s_5, s_6\}$. The lines defining the octagon induce a natural circular ordering on the sinks in the octagon set. For example, $s_1, s_6, s_{10}, s_8, s_9, s_7$ is the circular order of the octagon set of Fig. 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 $[1/2]$ Oct ($S$) consecutive sinks in Oct ($S$), called a reference set and denoted by $\text{REF}_i$, $i = 1, ..., |\text{Oct}(S)|$.
3) For each sink $p \in S$, compute the weight of $p$, equal to $\min_{e \in \text{REF}_i} d(p, r) + \max_{e \in \text{REF}_i} d(p, r)$.
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 $\text{dia}(S_1) + \text{dia}(S_2)$ is obtained.
6) Over all reference sets $\text{REF}_i$, select the partition ($S_1$, $S_2$) with smallest partition cost.

In the example of Fig. 12(a), each set of three consecutive sinks in the octagon set will be a possible reference set: $\text{REF}_1 = \{s_8, s_{10}, s_{16}\}$ has partition cost 280 as shown in Fig. 12(b); $\text{REF}_2 = \{s_{10}, s_{16}, s_{16}\}$ has partition cost 270 as shown in Fig. 12(c); etc. After all six reference sets have been evaluated, we find that the optimal reference set is $\text{REF}_1$, with cost 270. Fig. 12(d) shows the output of the BB + DME algorithm on the instance of Fig. 12(a).

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

\[^5\]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, [9] and [18] 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.

\[^6\]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.

---

**TABLE 1**

**DELAY AND CAPACITANCE AT EACH INTERNAL NODE IN ZST'S $T$ AND $T'$**

<table>
<thead>
<tr>
<th>$T$</th>
<th>$T'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>node</td>
<td>delay</td>
</tr>
<tr>
<td>$P_0$</td>
<td>50</td>
</tr>
<tr>
<td>$P_1$</td>
<td>64.0</td>
</tr>
<tr>
<td>$P_2$</td>
<td>67.3</td>
</tr>
<tr>
<td>$P_3$</td>
<td>71.9</td>
</tr>
<tr>
<td>$P_4$</td>
<td>77.6</td>
</tr>
<tr>
<td>$S_0$</td>
<td>454.0</td>
</tr>
</tbody>
</table>
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 $O(n|\text{Oct}(S)|)$ time. If $|\text{Oct}(S)| = \Theta(n)$, then the total time used in step 3 for a single bipartition can be reduced from $O(n^2)$ 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 $|\text{Oct}(S)| = \Theta(n)$ 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 $|\text{Oct}(S)| = O(1)$ 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" and $|\text{Oct}(S)| = \Theta(n)$, 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 $|\text{Oct}(S)| = O(1)$, the time complexity of BB is $O(n \log^2 n)$.

IV. 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 can 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 (1)–(5) used in [25] (the sizes of these examples range from 267 to 3,101 sinks).

4.1. Linear Delay Model

Our experimental results for linear delay are contained in Table II. 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.

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

---

Fig. 12. (a) Octagon lines of 16 sink locations. (b) Partition result of REF1. (c) Partition result of REF2. (d) The ZST produced by BB + DME.
TABLE II
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

<table>
<thead>
<tr>
<th>number of sinks</th>
<th>MMM cost</th>
<th>KCR cost</th>
<th>KCR + DME cost</th>
<th>reduction by KCR + DME from KCR (%)</th>
<th>reduction by BB + DME from KCR (%)</th>
<th>reduction by BB + DME from MMM (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary1</td>
<td>269</td>
<td>161.7</td>
<td>153.9</td>
<td>140.3</td>
<td>8.8</td>
<td>140.5</td>
</tr>
<tr>
<td>Primary2</td>
<td>603</td>
<td>403.6</td>
<td>376.7</td>
<td>350.4</td>
<td>7.0</td>
<td>360.8</td>
</tr>
<tr>
<td>r1</td>
<td>267</td>
<td>1,815</td>
<td>1,627</td>
<td>1,497</td>
<td>8.0</td>
<td>1,500</td>
</tr>
<tr>
<td>r2</td>
<td>598</td>
<td>3,625</td>
<td>3,349</td>
<td>3,013</td>
<td>10.0</td>
<td>3,010</td>
</tr>
<tr>
<td>r3</td>
<td>862</td>
<td>4,643</td>
<td>4,360</td>
<td>3,902</td>
<td>10.5</td>
<td>3,908</td>
</tr>
<tr>
<td>r4</td>
<td>1,903</td>
<td>9,276</td>
<td>8,580</td>
<td>7,782</td>
<td>9.3</td>
<td>8,000</td>
</tr>
<tr>
<td>r5</td>
<td>3,101</td>
<td>13,805</td>
<td>12,928</td>
<td>11,665</td>
<td>9.8</td>
<td>11,757</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td></td>
<td></td>
<td>9.1</td>
<td>14.9</td>
<td>8.2</td>
</tr>
</tbody>
</table>

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.

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 III. Again, these results indicate 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 (RST's) 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 clock skew in the ZST's 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 0.5 cm $\times$ 0.5 cm, 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 IV 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 ps while that of BB + DME was only 0.5 ps, a 93% reduction. Fig. 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 0.91 ns. By contrast, the MMM algorithm yielded a tree with cost 59256 $\mu$m and delay 0.94 ns for this case.

V. 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
TABLE III
COMPARISON OF BB + DME WITH OTHER ALGORITHMS IN THE ELMORE DELAY MODEL

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

* Results for Tsay's algorithm were obtained from Dr. Ren-Song Tsay and were not available for the Primary1 and Primary2 benchmarks.

TABLE IV
MEAN DELAY TIME AND CLOCK SKEW FOR RANDOM SINK SETS (TIME UNIT = PICOSECOND). THE RIGHT-MOST THREE COLUMNS DISPLAY RATIOS BETWEEN THE RESULTS OF BB + DME AND MMM

<table>
<thead>
<tr>
<th>#pts</th>
<th>max delay (MM)</th>
<th>min delay (MM)</th>
<th>clock skew (MM)</th>
<th>max delay (BB + DME)</th>
<th>min delay (BB + DME)</th>
<th>clock skew (BB + DME)</th>
<th>max delay (BB + DME/MM)</th>
<th>min delay (BB + DME/MM)</th>
<th>clock skew (BB + DME/MM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>769.3</td>
<td>763.2</td>
<td>6.1</td>
<td>746.6</td>
<td>746.2</td>
<td>0.4</td>
<td>0.970</td>
<td>0.978</td>
<td>0.07</td>
</tr>
<tr>
<td>16</td>
<td>801.8</td>
<td>797.0</td>
<td>4.8</td>
<td>784.0</td>
<td>782.6</td>
<td>0.7</td>
<td>0.977</td>
<td>0.980</td>
<td>0.15</td>
</tr>
<tr>
<td>24</td>
<td>856.6</td>
<td>826.2</td>
<td>10.4</td>
<td>808.7</td>
<td>806.3</td>
<td>0.4</td>
<td>0.967</td>
<td>0.978</td>
<td>0.04</td>
</tr>
<tr>
<td>32</td>
<td>863.5</td>
<td>855.6</td>
<td>7.9</td>
<td>837.3</td>
<td>836.5</td>
<td>0.8</td>
<td>0.970</td>
<td>0.978</td>
<td>0.10</td>
</tr>
<tr>
<td>40</td>
<td>885.6</td>
<td>876.3</td>
<td>9.3</td>
<td>857.0</td>
<td>856.5</td>
<td>0.5</td>
<td>0.968</td>
<td>0.977</td>
<td>0.05</td>
</tr>
<tr>
<td>48</td>
<td>908.9</td>
<td>896.4</td>
<td>12.5</td>
<td>876.0</td>
<td>876.3</td>
<td>0.5</td>
<td>0.965</td>
<td>0.978</td>
<td>0.04</td>
</tr>
<tr>
<td>56</td>
<td>926.2</td>
<td>914.4</td>
<td>11.8</td>
<td>891.2</td>
<td>889.7</td>
<td>0.5</td>
<td>0.961</td>
<td>0.973</td>
<td>0.04</td>
</tr>
<tr>
<td>64</td>
<td>940.6</td>
<td>930.1</td>
<td>10.5</td>
<td>910.7</td>
<td>910.2</td>
<td>0.5</td>
<td>0.968</td>
<td>0.979</td>
<td>0.05</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.968</td>
<td>0.978</td>
<td>0.07</td>
</tr>
</tbody>
</table>

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

reduce the power needed to drive the clock signal, thus improving system feasibility and reliability. Finally, wirerelength 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 generating phase (BB) with the embedding phase (DME).

The BB heuristic generates a connection topology by recursively dividing the set of sinks into two subsets with similar 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 use only horizontal or vertical cuts to partition the set of sinks.

The 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 branches. 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, in the linear delay model, DME yields optimal total wirelength for a given connection topology, 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 some 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 further investigations in this area to have fruitful applications, for both clock tree construction and the broader area of high-performance routing.

VI. REMARKS

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 [9], [18]. 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.

APPENDIX: PROOFS OF LEMMAS 1, 2, AND 3

Lemma 1: The intersection of two TRR’s, R1 and R2, 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.

Proof: Rotate the plane by 45° so that the boundaries of \(R_1\) and \(R_2\) are vertical and horizontal line segments (see Fig. 14). Let \(R'_1\) and \(R'_2\) be the two TRR’s after rotation with boundary lines given by:

- \(R'_1\): \(a_1 \leq a_2\) and \(b_1 \leq b_2\)
  
  \[
  \begin{align*}
  x &= a_1 \\
  y &= b_1.
  \end{align*}
  \]

- \(R'_2\): \(a_3 \leq a_4\) and \(b_3 \leq b_4\)
  
  \[
  \begin{align*}
  x &= a_3 \\
  y &= b_3.
  \end{align*}
  \]

Then \(R'_1 \cap R'_2\) is a rectangular region with boundary lines

\[
\begin{align*}
  x &= \max(a_1, a_3) \\
  x &= \min(a_2, a_4) \\
  y &= \max(b_1, b_3) \\
  y &= \min(b_2, b_4).
\end{align*}
\]

Since rotating each TRR by 45° requires constant time, determining the intersection of the two TRR’s \(R_1 \cap R_2\) also requires only constant time.

If \(\text{radius}(R_1) + \text{radius}(R_2) = d(\text{core}(R_1), \text{core}(R_2))\), 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 \(\text{core}(R_1)\) and \(\text{core}(R_2)\) with length less than \(d(\text{core}(R_1), \text{core}(R_2))\). 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.

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 \(x\). Fig. 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 Lemma 2. 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

\[
PA(p, ms) \cup PA(q, ms) = \mathbb{R}^2.
\]

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 \(x\), \(PA(x, (x)) = \mathbb{R}^2\), this implies that either \(PA(p, ms) = \mathbb{R}^2\) or \(PA(q, ms) = \mathbb{R}^2\) 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 TRR’s, \(rr_p\) and \(rr_q\), such that \(core(rr_p) = (p), core(rr_q) = (q)\), and

\[
\begin{align*}
\text{radius} (rr_p) &= x \cdot d(p, q) \\
\text{radius} (rr_q) &= (1 - x) \cdot d(p, q)
\end{align*}
\]

for some \(x\) satisfying \(0 \leq x \leq 1\). If \(x = 1\) or \(x = 0\), the lemma is immediately true, since either \(PA(p, (p)) = \mathbb{R}^2\) or \(PA(q, (q)) = \mathbb{R}^2\) 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 Fig. 16:

a) \(z_1\) and \(z_2\) are both corners of the same TRR, either \(rr_p\) or \(rr_q\). Assume without loss of generality that they are both corners of \(rr_p\);

b) \(z_1\) and \(z_2\) are corners of different TRR’s. Assume without loss of generality that \(z_1\) is a corner of \(rr_q\) and \(z_2\) is a corner of \(rr_p\).

Define a ray \(p_1, p_2\) from point \(p\) 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 Fig. 16(a) it is evident that \(PA(p, (z_i))\) is a half plane with border line \(z_i, p_1\) and \(PA(p, (z_i))\) is a half plane bordered by line \(z_i, p_2\). Furthermore, \(PA(p, ms)\) is the infinite region separated from \(p\) by (and including) ray \(z_1, p_1\), segment \(ms\), and ray \(z_2, p_2\). 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 Fig. 16(b), \(PA(p, ms)\) is the infinite region separated from \(p\) by (and including) \(z_1, p_1\), \(ms\), and \(z_2, p_2\). \(PA(q, ms)\) is the region separated from \(q\) by the same border. Again, \(PA(p, ms) \cup PA(q, ms) = \mathbb{R}^2\).

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 Fig. 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 \(a\) to \(q\). Because \(aq'q\) is a straight-line path, segment \(aq\) can be replaced by segments \(aq'\) 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 \((T, x)\) 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 \(cost(T') = cost(T) - length(T', q'q)\).

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 \(d(a, b) > \)
Because \( d(a, b) > d(ms(a), ms(b)) \) and \( d(a, b) > |l_{LD}(a) - l_{LD}(b)| \), \( d(a, b) \) is strictly greater than the merging cost between \( ms(a) \) and \( ms(b) \). Therefore,
\[
|e_q| = |e_{q'}| + |e_p| \quad \text{and} \quad |e_q| = |e_p| + |e_q|.
\]

Equations (3) and (4) imply that \( |e_q| > 0 \), and thus
\[
|e_q| + |e_q| > |e_q| + |e_q| + |e_q|.
\]

As a result, \( \text{cost}(T') < \text{cost}(T) \).

\[\square\]

**ACKNOWLEDGMENT**

The authors are grateful to Dr. Ren-Song Tsay for providing benchmark data and for his communications which made this collaboration possible. We also thank Mr. Masato Edahiro for pointing out a mistake in our original of Lemma 3.

**REFERENCES**


Vincent Hsu received the B.S. degree in mathematics from Cheng Kung University, Taiwan, in 1971, the M.S. degree in mathematics, and the Ph.D. degree in computer science from Tsing Hua University, Hsinchu, Taiwan in 1975 and 1992, respectively. He is currently an Associate Professor of Computer Science at the University of California, Riverside. His research interests include computer-aided design for VLSI, computer architecture design, and synthesized circuits. He co-received an Outstanding Young Author Award from IEEE Circuits and Systems Society in 1990. Dr. Hsu is a member of ACM.

Jan-Ming Ho received the B.S. degree from Cheng-Kung University, Taiwan, in 1978, the M.S. degree from Chiou-Tong University, Taiwan, in 1980, both in electrical engineering, and the Ph.D. degree in computer science from Northwestern University, in 1989. Currently, he is an Associate Research Fellow at the Academia Sinica, Taiwan. His major research area is in studying combinatorial algorithms for computer-aided design and their implementations. His area of interest also includes operating systems and computer architecture designs.

Kenneth D. Boese (S'92) is pursuing the Ph.D. degree in computer science at UCLA, where he holds a GTE Graduate Fellowship. He received the A.B. degree in Mathematics from Princeton University. His research interests include VLSI global routing, partitioning and clustering, cost surfaces in stochastic optimization, and Bayesian networks.

Mr. Boese is a member of ACM.

Yu-Chin Hsu received the B.S. degree in computer science and information engineering from National Taiwan University, Taiwan, in 1981, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1986 and 1987, respectively. From 1987 to 1991, he was an associate professor of computer science at Tsing Hua University, Hsin-Chu, Taiwan. He is currently Associate Professor of computer science at the University of California, Riverside. His research interests include most aspects of computer-aided design for VLSI, computer architecture design, and synthesis. Dr. Hsu is a member of ACM.

Andrew B. Kahng (AM'89) received the A.B. degree in applied mathematics and physics from Harvard College, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego. He has been an assistant professor in the computer science department at UCLA since July 1989. His research interests include computer-aided design of VLSI circuits, combinatorial and parallel algorithms, global optimization theory, and computational geometry.

Dr. Kahng is a member of ACM.