Planar-DME: A Single-Layer Zero-Skew Clock Tree Router *

Andrew B. Kahng and Chung-Wen Albert Tsao

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

Abstract

This paper presents new single-layer, i.e., planar-embeddable, clock tree constructions with exact zero skew under either the linear or the Elmore delay model. Our method, called Planar-DME, consists of two parts. The first algorithm, called Linear-Planar-DME, guarantees an optimal planar zero-skew clock tree (ZST) under the linear delay model. The second algorithm, called Elmore-Planar-DME, uses the Linear-Planar-DME connection topology in constructing a low-cost ZST according to the Elmore delay model. While a planar ZST under the linear delay model is easily converted to a planar ZST under the Elmore model by elongating tree edges in bottom-up order, our key idea is to avoid unneeded wire elongation by iterating the DME construction of ZST and the bottom-up modification of the resulting non-planar routing. Costs of our planar ZST solutions are comparable to those of the best previous non-planar ZST solutions, and substantially improve over previous planar clock routing methods.

1 Preliminaries

The placement phase of physical layout determines positions for the synchronizing elements of a circuit, which are the *sinks* of the clock net. Large cell-based designs can have thousands of sinks in a clock net, with these sinks located quite arbitrarily throughout the layout region. We denote the set of sink locations in a clock routing instance as $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$. A *connection topology* is a rooted binary tree, G, with n leaves corresponding to the sinks in S. A *clock tree* $T(S)$ is an embedding of the connection topology in the Manhattan plane, i.e., a placement in $\Re^2$ which assigns each internal node $v \in G$ to a location which we denote as $l(T, v)$, or more simply as $l(v)$ when no confusion arises. The root of the clock tree is the *source*, denoted by s_0. When

*Partial support for this work was provided by NSF MIP-9223740 and Young Investigator Award MIP-9257982.
The authors are reachable via http://ballade.cs.ucla.edu:8080/~abk and http://ballade.cs.ucla.edu:8080/~tsao.

the clock tree is rooted at the source, any edge between a parent node p and its child v may be identified with the child node, i.e., we denote this edge as e_v. In our discussion, the *distance* between two points p and q is the Manhattan distance $d(p, q)$, and the distance between two sets of points P and Q, $d(P, Q)$, is $min\{d(p, q) \mid p \in P \text{ and } q \in Q\}$. The *cost* of the edge e_v is simply its wirelength, denoted $|e_v|$; this is always at least as large as the Manhattan distance between the endpoints of the edge, i.e., $|e_v| \geq d(l(p), l(v))$. The cost of $T(S)$ is the total wirelength of the edges in $T(S)$.

For a given clock tree $T(S)$, let $t(s_0, s_i)$ denote the signal propagation time on the unique path from source s_0 to sink s_i. The *skew* of $T(S)$ is the maximum value of $|t(s_0, s_i) - t(s_0, s_j)|$ over all sink pairs $s_i, s_j \in S$. If the skew of $T(S)$ is zero then $T(S)$ is a *zero-skew tree* (ZST). In what follows, we address the

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

1.1 Minimum-Cost Zero-Skew Trees

In the IC CAD literature, minimum-cost, exact zero-skew clock trees for cell-based designs have been constructed by applying geometric optimizations over the set of sink locations. The associated formulations are perhaps best motivated by the "monolithic" single-buffer clocking approach [2, 10]. From the circuits/systems perspective, workers such as Friedman [14] have considered these geometric methods as "subroutines" in addressing further concerns such as use of existing distributed buffers, non-zero clocking skew, driver and buffer sizing, etc. Thus, the zero-skew clock routing problem, along with its planar variant, remains a fundamental building block in any clock distribution methodology.

The first clock tree construction for cell-based layouts with arbitrary sink locations was that of Jackson et al. [16]; their MMM algorithm does recursive top-down partitioning of the set of sinks into two equal-sized subsets, always connecting the centroid of a set to the centroids of its subsets. Kahng et al. [8, 18] constructed clock tree topologies using a bottom-up matching approach. Their "KCR" algorithm obtains zero pathlength skew in practice (i.e., zero skew under the linear delay model) but has no theoretical guarantee. The work of Tsay [23] was the first to guarantee exact zero skew for any input; this was accomplished with respect to the Elmore delay model. In the same spirit as [18], Tsay recursively combines pairs of zero skew trees at "tapping points" to yield larger zero skew trees. To maintain the exact zero skew, wires are elongated via "snaking" as necessary.

The above methods all concentrate on generation of the clock tree *topology*: the topology is then embedded in the plane more or less arbitrarily as it is generated. The Deferred-Merge Embedding

(DME) method, which was discovered independently by three groups [3, 4, 11], is a linear-time algorithm which optimally embeds any given topology in the Manhattan plane, i.e., with exact zero skew and minimum total wirelength. Because the properties of the DME construction are central to our present work, we now provide a review of DME following the development in [3].

The DME Algorithm

Given sink set S and topology G, DME embeds internal nodes of G via: (i) a bottom-up phase that constructs a tree of *merging segments* which represent loci of possible placements of internal nodes in the ZST T; and (ii) a top-down embedding phase that determines exact locations for the internal nodes in G (see Figure 3, reproduced from [3]).

In the **bottom-up phase** (Figure 3a), each node $v \in G$ is associated with a merging segment which represents a set of possible placements of v in a minimum-cost ZST. The merging segment of a node depends on the merging segments of its two children, hence the bottom-up processing order. More precisely, let a and b be the children of node v in G, and let TS_a and TS_b denote the subtrees of merging segments rooted at a and b, respectively. We seek placements of v which allow TS_a and TS_b to be merged with *minimum* added wire while preserving zero skew. This means that we want to minimize $|e_a| + |e_b|$ in T, while balancing delays from $l(v)$ to all leaves in the subtree rooted at v. The values of $|e_a|$ and $|e_b|$ which achieve this are unique; they are computed and stored for use in the top-down embedding phase of DME.

To formally describe this construction, the following terminology is useful. 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 (Figure 1, reproduced from [3]). The Manhattan arc at the center of the TRR is called its *core*. Finally, the *radius* of a TRR is the distance between its core and its boundary. Note that a Manhattan arc is itself a TRR with radius 0.

A formal recursive definition of $ms(v)$, the merging segment of node $v \in G$, is as follows. If v is a sink s_i, then $ms(v) = \{s_i\}$ (note that this single point is a Manhattan arc). If v is an internal node, then $ms(v)$ is the set of all placements $l(v)$ which merge TS_a and TS_b with minimum wire cost, i.e., 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 $ms(v) = trr_a \cap trr_b$ is obtained by intersecting two TRRs, trr_a with core $ms(a)$ and radius $|e_a|$, and trr_b with core $ms(b)$ and radius $|e_b|$ (see Figure 2, also reproduced from [3]). Boese and Kahng [3] show that if $ms(a)$ and $ms(b)$ are both Manhattan arcs, then $ms(v)$ is also a Manhattan arc. Since the merging segment $ms(s_i)$ for each sink s_i is a single point and thus a Manhattan arc, by induction all merging segments are Manhattan arcs.

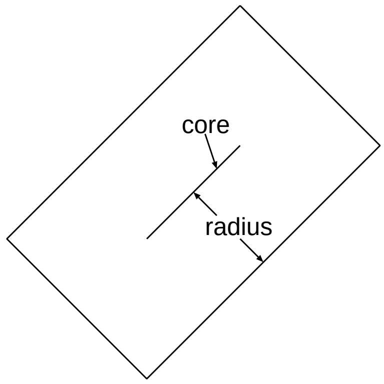


Figure 1: An example of a TRR.

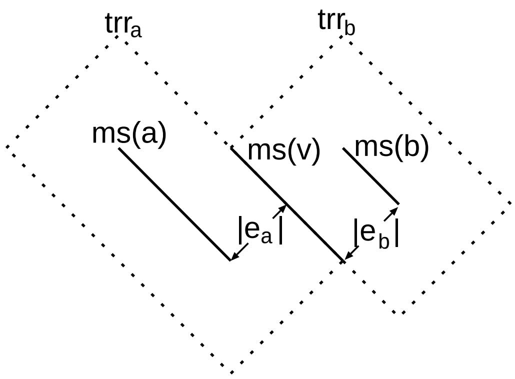


Figure 2: Construction of merging segment $ms(v)$.

Given the tree of merging segments corresponding to G, the **top-down phase** (Figure 3b) chooses exact embeddings of internal nodes in the ZST as follows. For node v in topology G, (i) if v is the root node, then DME selects any point in $ms(v)$ to be $l(v)$;[1] or (ii) if v is an internal node other than the root, DME chooses $l(v)$ to be any point in $ms(v)$ that is at distance $|e_v|$ or less from the placement of v's parent p (the merging segment $ms(p)$ was constructed such that $d(ms(v), ms(p)) \leq |e_v|$, so there must exist some $l(v)$ satisfying this condition). In case (ii), $l(v)$ can be any point in the intersection of $ms(v)$ and the square TRR trr_p which has radius $|e_v|$ and core $\{l(p)\}$.

Note that DME requires an input topology. Several works have thus proposed topology constructions that yield low-cost routing solutions when DME is applied. Below, we compare against the non-planar KCR+DME [3] and Greedy-DME [12] methods.

[1] If a fixed clock source location clk has been specified, DME chooses $l(s_0) \in ms(s_0)$ with minimum distance from clk and connects a wire directly from clk to $l(s_0)$.

Procedure Build_Tree_of_Segments (G,S)

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,
    $ms(v) \leftarrow \{l(v)\}$
  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:
      $core(trr_a) \leftarrow ms(a)$
      $radius(trr_a) \leftarrow |e_a|$
      $core(trr_b) \leftarrow ms(b)$
      $radius(trr_b) \leftarrow |e_b|$
    $ms(v) \leftarrow trr_a \cap trr_b$

(a) Bottom-up Phase: Construction of the tree of merging segments TS.

Procedure Find_Exact_Placements(TS)

Input: Tree of segments TS containing $ms(v)$, and value of $|e_v|$ 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 $l(v) \in ms(v)$
  else
    Let p be the parent node of v
    Construct trr_p as follows:
      $core(trr_p) \leftarrow \{l(p)\}$
      $radius(trr_p) \leftarrow |e_v|$
    Choose any $l(v) \in ms(v) \cap trr_p$

(b) Top-down Phase: Construction of the ZST by embedding internal nodes of G within TS.

Figure 3: The DME algorithm. The procedure Calculate_Edge_Lengths in (a) finds the values $|e_a|$ and $|e_b|$ such that $|e_a| + |e_b|$ is minimized and zero-skew is achieved; this calculation depends on the delay model.

1.2 Planar-Embeddable Trees

None of the clock tree solutions given by the above "exact zero skew" algorithms is easily embedded in the layout plane: it is often impossible to perform the actual layout without introducing many vias. This difficulty was first noted by Zhu and Dai [25], who stated compelling reasons to seek a *single-layer*, or *planar-embeddable* clock routing solution.

- The clock routing layer may be prescribed, or we may prefer the layer with smallest RC delay.

- Routing on fewer distinct layers (i.e., having fewer distinct electrical parameters) makes the layout more independent of process variation. Uniform electrical parameters also simplify buffering optimizations.

- Single-layer routing eliminates the delay and attenuation of the clock signal through vias, thus improving both performance and signal integrity.

Given these observations, the **Planar Zero-Skew Clock Routing** problem is of interest, i.e., given sink set S, find a *planar-embeddable* ZST $T(S)$ with minimum cost.

"Planar-embeddable" intuitively means that the tree "can be drawn in the plane without edges crossing", but this concept is not easily characterized in the Manhattan plane. Existing work [25] implicitly relies on Euclidean planar-embeddability being sufficient for Manhattan planar-embeddability (a line segment in the Euclidean plane can be approximated to any desired accuracy by a monotone staircase in the Manhattan plane). Thus, we define two edges as crossing each other when the corresponding *open* line segments in the Euclidean plane properly intersect (share exactly one point). This definition allows optimal planar clock routing solutions where the embeddings of edges are superposed. Figure 4 shows this phenomenon: four sinks that are collinear will have an optimal "planar" clock tree whose edges pass over each other. Since this overlapping can be made planar with minimum increase in wirelength, we accept such a degenerate solution as planar. This is also the convention of [25].

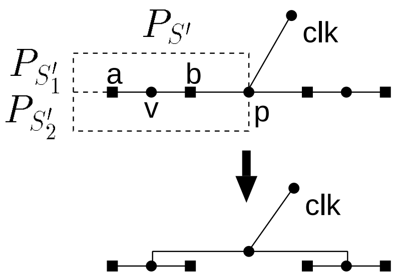


Figure 4: For these four sinks on a line, edges of the optimal planar ZST will overlap. We accept this since the ZST can be made non-overlapping with minimal increase in wirelength. (The convex polygons $P_{S'}$, $P_{S'_1}$, and $P_{S'_2}$ and the points p and v pertain to the later discussion about the partitioning rules for Linear-Planar-DME.)

The clock routing method of Zhu and Dai [25] was the first to guarantee a planar ZST; the solution has minimum possible source-sink pathlength, and runs in between $\Omega(n \log n)$ and $O(n^2)$ time. However, the method basically creates an "X" clock tree, where an "H" would considerably reduce the tree cost. Khan et al. [19] have observed this deficiency, and have proposed a guaranteed-planar heuristic which reduces the tree cost by applying the top-down horizontal/vertical partitioning approach of [16] for a user-specified number of levels, then applying the Zhu-Dai X-tree construction within each of the resulting regions. When the user-specified number of levels is zero, the method reduces to that of Zhu-Dai. The authors of [19] claim that their algorithm guarantees minimum source-sink pathlength delay, and report approximately 10% wirelength reduction over [25]. Both [19, 25] rely completely on the linear delay model to achieve their results.

1.3 Organization of Paper

The remainder of this paper is organized as follows. Section 2 shows that under the linear delay model, the two passes (bottom-up and top-down) of the DME algorithm can be replaced by a *single* top-down pass which yields exactly the same (optimal) solution. From this "Single-Pass DME" method, we develop a version called Linear-Planar-DME which guarantees a planar, optimal ZST solution under the linear delay. Because Single-Pass DME cannot be applied to the Elmore delay model, Section 3 presents the Elmore-Planar-DME heuristic, which can transform the Linear-Planar-DME solution to a planar Elmore-ZST with little cost increase. Section 4 discusses Linear-Planar-DME variants which can produce good input topologies for Elmore-Planar-DME, and Section 5 gives experimental results and comparisons with previous work. We conclude in Section 6 by listing several extensions and directions for future work.

2 The Linear-Planar-DME Algorithm

Describing our new planar clock routing algorithm requires a little more terminology. For any sink subset $S' \subseteq S$, we define the *diameter* of S' as $diam(S') = \max\{d(s_i, s_j) | s_i, s_j \in S'\}$. The *radius* of S' is then $radius(S') = diam(S')/2$. A *Manhattan disk* is a TRR with a core consisting of a single point; we use $MD(s_i, r)$ to denote the Manhattan disk with core $\{s_i\}$ and radius $r \geq 0$. In other words, a Manhattan disk is the "diamond-shaped" set of all points within a prescribed radius of a central point. For any sink set $S' \in S$ with $radius(S') = r'$, we define $center(S') = \bigcap_{s_i \in S'} MD(s_i, r')$, which is so named because the the distance from $center(S')$ to any sink in S' is at most r'. We use $c(S')$ to denote the midpoint of $center(S')$.

Finally, we use two terms that are defined in the Euclidean plane: (i) $P_{S'}$ denotes any Euclidean *convex polygon* containing S', and (ii) *convex-hull*(S') is the $P_{S'}$ with minimum interior area. We say that a point p lies *inside* $P_{S'}$ if p is on the boundary of $P_{S'}$ or is strictly interior to $P_{S'}$. These terms will be used in proving that the Linear-Planar-DME algorithm defined below yields a planar solution: wires embedded along the boundary between two disjoint (Euclidean) convex polygons cannot intersect subsequent wires embedded internally to these polygons.

2.1 Single-Pass DME

Our first theoretical result is that under the linear delay model, a single top-down phase can yield the same output as the original two-phase DME algorithm. We prove that the tree of merging segments constructed in the bottom-up phase can be generated *during* the top-down phase. This result follows from properties of the minimum-pathlength zero-skew subtree over any sink set S', notably that the root of the subtree over S' must be located at $center(S')$. More precisely, $center(S')$ is equal to $ms(v)$, where v is the root of the tree of merging segments constructed by DME over S', and $ms(v)$ is hence independent of tree topology. This leads to what we call the Single-Pass DME algorithm.

The following Facts and Theorem are crucial to the development of the Single-Pass DME and then the Linear-Planar-DME algorithms. Two useful facts are due to [3]. Fact 1 is a straightforward extension of Theorem 2 in [3][2], and Fact 2 is proved in the analysis of the same Theorem 2.

Fact 1: For any sink set S and topology G, let S_v be the set of sinks in the subtree rooted at v in the DME solution. Let $t_{LD}(u)$ be the linear delay (i.e., pathlength) from point $u \in ms(v)$ to each sink in S_v. Then $t_{LD}(u) = radius(S_v)$. □

Fact 2: For any sink set S and topology G, let $r = radius(S)$ and let $TRR(v)$ denote the special tilted rectangular region that corresponds to either $TRR(v) = MD(l(v), r)$ if v is a sink node, or $TRR(v) = TRR(a) \cap TRR(b)$ if v is an internal node of G with children a and b. Then for each node $v \in G$, $ms(v) = core(TRR(v))$. □

Fact 3: For any sink subset $S' \subseteq S$, $diam(S')$ can be computed in linear time.

Proof: Under a 45-degree rotation of the coordinate axes (and scaling by a $\sqrt{2}$ factor), Manhattan distance is transformed into L_∞ distance. Such a rotation can be accomplished in constant time per point in S'. Let $x_{max}(y_{max})$ and $x_{min}(y_{min})$ respectively denote the largest and smallest x-coordinates (y-coordinates) among all the points in the rotated S'. Then $diam(S') = \max(x_{max} - x_{min}, y_{max} - y_{min})$ and is found in $O(|S'|)$ time. □

Theorem 1: Given a set of n sinks $S \in \Re^2$ and connection topology G, we can produce the same output ZST $T(S)$ that the DME algorithm will produce under the linear delay model, using only a single top-down phase with time complexity between $\Omega(n \log n)$ and $O(n^2)$.

Proof: We determine the merging segments and incident edge lengths for all nodes in top-down

[2] Theorem 2 in [3] states that for any sink set S and topology G, the DME algorithm will find a ZST with source-sink pathlength delay $T_{LD}(s_0) = diam(S)/2$.

order as follows. Let v be a node in G with parent p (if v is not the root). As in the statement of Fact 1, let S_v and S_p be the sets of sinks in the subtrees rooted at nodes v and p in the DME solution. For any sink subset $S_v \subseteq S$, the value of $r' = radius(S')$ can be found in $\Theta(|S_v|)$ time (Fact 3), and in $\Theta(|S_v|)$ time we can build $MD(l(u), r')$ for all sinks $u \in S_v$.

Let $TRR(v)$ be defined as in the statement of Fact 2, then recursive application of Fact 2 shows that $ms(v) = core(TRR(v)) = core(\bigcap_{u \in S_v} TRR(u)) = core(\bigcap_{u \in S_v} MD(l(u), r))$. From Lemmas 1, 2 and 3 in [3], we have $ms(v) = core(\bigcap_{u \in S_v} MD(l(u), r') = \bigcap_{u \in S_v} MD(l(u), r') = center(S_v)$, where $r' = radius(S_v)$. Since the intersection of any two TRR's can be found in constant time and is also a TRR, we can compute $ms(v)$ in time $\Theta(|S_v|)$.

By Fact 1, the length of the edge incident to node v in G, $|e_v|$, is equal to $t_{LD}(p) - t_{LD}(v) = radius(S_p) - radius(S_v)$. By Fact 3, we can compute $|e_v|$ for node v in time $\Theta(|S_v|)$ since we already have $t_{LD}(p) = radius(S_p)$. Thus, we can compute $ms(v)$ and $|e_v|$ in $\Theta(|S_v|)$ time, and we now have all the information that would have been determined in the bottom-up phase of DME, and the single top-down phase is sufficient. Finally, let L_i denote the set of nodes at level i of the ZST, and let l be the height of the ZST. We have $\sum_{v \in L_i} |S_v| \leq n$, and $\log n \leq l \leq n$. Thus, the overall time complexity is $l \times \sum_{v \in L_i} |S_v| \leq ln = \Theta(ln)$, which is between $\Omega(n \log n)$ and $O(n^2)$. □

Because Single-Pass DME outputs the same optimum ZST as the original DME with respect to the given connection topology, established properties of the output tree (minimum source-sink pathlength, and minimum total tree cost) are maintained. The worst-case and best-case time bounds are the same as those for the method of [25].

2.2 Linear-Planar-DME

The impact of Theorem 1 may not be immediately apparent, since DME can already accomplish the same construction in linear time. However, the proof showed that as soon as Single-Pass DME has been given a partitioning of S_v into S_a and S_b, it can immediately find the $ms(a)$ and $ms(b)$ that are compatible with an optimal ZST having this "top part" of the clock topology. Thus, Single-Pass DME allows the connection topology to be determined dynamically in a top-down fashion, yet still finds a minimum-pathlength, minimum-cost embedding of whatever topology is eventually determined. If Single-Pass DME chooses and embeds the connection topology carefully, then a *planar* routing can be achieved.

The Linear-Planar-DME algorithm is essentially a version of Single-Pass DME wherein the connection topology is determined based on the existing routing, such that future routing cannot interfere with this existing routing. We use the (Euclidean) *convex polygon* concept to guide the top-down partitioning of both the routing area and the set of sinks. Given $S' \subseteq S$ and a convex polygon $P_{S'}$ containing S', we recursively divide $P_{S'}$ into two smaller convex polygons such that routing inside each smaller polygon cannot interfere with routing inside the other polygon or on the boundary between the polygons.

Embedding and Partitioning Rules

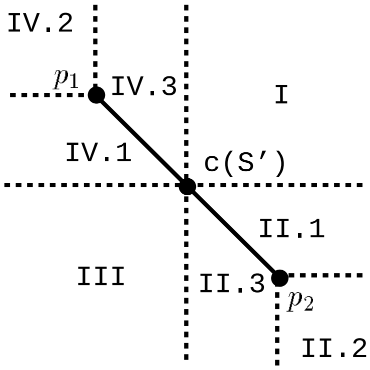
The Linear-Planar-DME algorithm is derived from Single-Pass DME by adopting the following rules for embedding the internal nodes of the ZST, and for top-down bipartitioning of the sinks in each subtree.

- **Embedding Rules**: In each recursive call, Linear-Planar-DME accepts a subset of sinks $S' \subseteq S$, some convex polygon $P_{S'}$ containing S', and some point p inside $P_{S'}$ which is to connect to a point v on $ms(v) = center(S')$. The point p is the embedding of the parent of the root of the subtree over S'; this point has already been determined earlier in the top-down pass.[3] The existing routing is outside $P_{S'}$, hence if we can select a feasible DME embedding point v inside $P_{S'}$, the routing from p to v will not interfere with the routing external to $P_{S'}$. Thus, the resulting routing will be planar and compatible with the DME solution. The embedding rules in Figure 5 ensure that such an embedding point will be selected in $O(1)$ time.

- **Partitioning Rules**: Our goal is to find a *splitting line* which (i) divides $P_{S'}$ into two convex polygons $P_{S'_1}$ and $P_{S'_2}$ and thus also partitions the sink set between the two subtrees of v, and (ii) allows the routing from p to v to be on the boundary between $P_{S'_1}$ and $P_{S'_2}$. The rules to determine the splitting line for S' are shown in Figure 5. Sinks lying inside one of the convex polygons are assigned to that polygon to determine sets S'_1 and S'_2; a sink on $\overleftrightarrow{pv}$ can be assigned to either polygon, so long as neither S'_1 or S'_2 is empty. In the example of Figure 4, $S' = \{a, b\}$ is divided into $S'_1 = \{a\}$ and $S'_2 = \{b\}$, and $P_{S'}$ is divided into $P_{S'_1}$ and $P_{S'_2}$ accordingly. A total of $\Theta(|S'|)$ time is needed to partition the sinks in set S'.

The overall Linear-Planar-DME algorithm is given in Figure 6. Steps 4 and 6 in Linear-Planar-DME-Sub are the key difference between Linear-Planar-DME and generic Single-Pass DME. Single-Pass DME would more or less arbitrarily choose a feasible embedding point at Step 4, and partition the sinks in the subtree according to the given connection topology at Step 6. In contrast, Linear-Planar-DME chooses both the embedding and the partition of the sinks (thus dynamically determining the topology) so that planarity is maintained. If the clock source location is not specified, then Linear-Planar-DME will set the root of the clock tree to be the clock location. Figure 7 illustrates how the planar clock routing is achieved by Linear-Planar-DME. For any given sink set, applying the partitioning and embedding rules takes the same (linear) time

[3] When the meaning is clear, our discussion will use, say, v to denote either a node in the tree topology, or the point at which that node has been embedded in the Manhattan plane (that is to say, $l(v)$).



The Embedding Rules

	Location of p	Embedding point for node v
A1	Regions I, III	$c(S')$
A2	Regions II.1, IV.1	intersection of $\overline{p_1p_2}$ with horizontal line through p
A3	Regions II.3, IV.3	intersection of $\overline{p_1p_2}$ with vertical line through p
A4	Region II.2 (IV.2)	p_2 (p_1)

The Partitioning Rules

	Locations of p, v	Splitting line
B1	$p \neq v$	$\overleftrightarrow{pv}$
B2	$p = v$; $p \neq c(S')$	$\overleftrightarrow{p_1p_2}$
B3	$p = v$; $p = c(S')$	Vertical line through p

Figure 5: Rules used by Linear-Planar-DME: (i) to choose the embedding point of v (the root of the subtree over sink set $S' \subseteq S$ in *any* DME solution), and (ii) to choose the splitting line to partition the sink set S' based on the relative positions of v's parent p and $center(S') = \overline{p_1p_2}$. Without loss of generality, assume that the Manhattan arc $center(S')$ has slope -1. Let the coordinates of $c(S')$, p_1 and p_2 be (x_c, y_c), (x_1, y_1), and (x_2, y_2), respectively. We define: Region *I*: $x \geq x_c$, $y \geq y_c$; Region *II*.1: $y \geq -x + y_1 + x_1$, $y \leq y_c$, $y \geq y_2$; Region *II*.2: $x \geq x_2$, $y \leq y_2$; and Region *II*.3: $y \leq -x + y_1 + x_1$, $x \geq x_c$, $x \leq x_1$. Regions *III*, *IV*.1, *IV*.2, and *IV*.3 are defined similarly.

that is required to compute merging segments and edge lengths, hence Linear-Planar-DME has the same time complexity as Single-Pass DME.

2.3 Correctness of Linear-Planar-DME

We now show that Linear-Planar-DME yields a planar-embeddable, i.e., single-layer optimal ZST. The following Fact 4 states that for any sink set $S' \subseteq S$, the midpoint of $center(S')$ must lie inside *convex-hull*(S'). Note that $center(S')$ does not necessarily lie entirely in *convex-hull*(S') – consider the case of S' containing two points along a diagonal line.

Algorithm Linear-Planar-DME (S,clk)
Input: Set of sinks S; clock location clk in P_S.
Output: Planar ZST $T(S)$ with root s_0; $cost(T)$.
$r \leftarrow radius(S)$; Build $MD(l(s_i), r)$ for all sinks $s_i \in S$; $center(S) \leftarrow \bigcap_{s_i \in S} MD(l(s_i), r)$; **if** clk not specified Embed s_0 at $c(S)$ (i.e., $l(s_0) \leftarrow c(S)$); **else** Embed s_0 at clk (i.e., $l(s_0) \leftarrow clk$); $t_{LD}(s_0) \leftarrow r + d(l(s_0), center(S))$; $P_S \leftarrow$ a rectangle containing S and clk; Linear-Planar-DME-Sub(S,P_S,s_0) ; $cost(T) \leftarrow \sum_{v \in T, v \neq s_0} \lvert e_v \rvert$;

Procedure Linear-Planar-DME-Sub (S',$P_{S'}$,p)
Input: Set of sinks $S' \subseteq S$; convex polygon $P_{S'}$ containing S'; parent node p lying inside $P_{S'}$.
Output: Planar ZST $T(S')$ with root v.
$t_{LD}(v) = r' \leftarrow radius(S')$; $ms(v) = center(S') \leftarrow \bigcap_{s_i \in S'} MD(l(s_i), r')$; $\lvert e_v \rvert \leftarrow t_{LD}(p) - t_{LD}(v)$; Embed node v at $l(v) \in ms(v)$ by embedding rules in Figure 5; Connect a wire from $l(p)$ to $l(v)$; Divide S' and $P_{S'}$ into S'_1, S'_2 and $P_{S'_1}$, $P_{S'_2}$ by partitioning rules in Figure 5; $parent(v) \leftarrow p$; **if** $\lvert S' \rvert = 1$ **return**; Linear-Planar-DME-Sub(S'_1,$P_{S'_1}$,v); Linear-Planar-DME-Sub(S'_2,$P_{S'_2}$,v);

Figure 6: The Linear-Planar-DME Algorithm.

Fact 4: For any $S' \subseteq S$, $c(S')$ lies inside *convex-hull*(S').

Proof: Without loss of generality, assume that the Manhattan arc $center(S') = \overline{p_1 p_2}$ has slope -1, as shown in Figure 8, where $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, with $x_1 < x_2$ and $y_2 < y_1$. Let $radius(S') = r'$ and $R = MD(p_1, r') \cap MD(p_2, r')$. Since $d(p_1, p_2) \leq diam(S') = 2r'$, R is nonempty and has the following vertices: $A = (x_1 - \delta, y_2), B = (x_1, y_2 - \delta), C = (x_2, y_1 + \delta)$, and $D = (x_2 + \delta, y_1)$ where $\delta = r' - d(p_1, p_2)/2 \geq 0$. Since any sink in S' must be located within distance r' from p_1 and p_2, $S' \subseteq R$. Since no sinks of S' lie outside R, $\overline{AB}$ and $\overline{CD}$ must each contain a sink of S'; otherwise, $center(S')$ would become a TRR with non-zero width. Similarly, $\overline{AC}$ and $\overline{BD}$ must each contain a sink of S'; otherwise, $center(S')$ would extend into region $IV.2$ or $II.2$ in the Figure. Thus we can assume that there are sinks x, y, s, and t lying on $\overline{AB}$, $\overline{CD}$, $\overline{AC}$ and $\overline{BD}$, respectively (these correspond to between two and four distinct points, e.g., x and

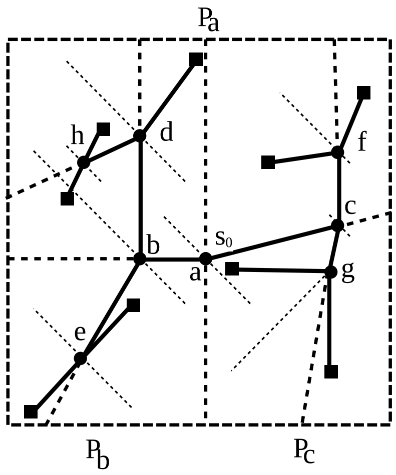


Figure 7: Example with 9 sinks (squares at leaf nodes in tree), illustrating the execution of Linear-Planar-DME. The entire routing region (P_a) is recursively divided into convex polygons (boundaries of polygons indicated by thick dashed lines and tree edges) until only one sink lies within each convex polygon. The tree of merging segments is given by thin dotted lines. The root of the clock tree, s_0, is embedded as specified by Linear-Planar-DME in Figure 6. The internal nodes $a, b, \cdots, h$ are embedded by Rules $A1$, $A2$, $A1$, $A3$, $A1$, $A3$, $A4$, and $A1$. Note that $s_0 = a$. Polygon P_a is divided by the vertical line through a (Rule $B3$) into polygons P_b and P_c which contain sink sets rooted at nodes b and c. All other partitioning steps invoke Rule $B1$ only.

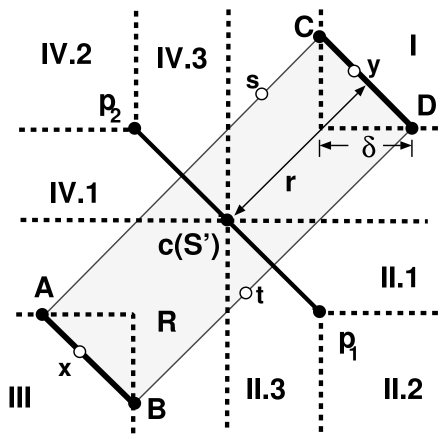


Figure 8: For any sink set $S' \subseteq S$ with $radius(S') = r'$, the mid-point of $center(S') = \overline{p_1 p_2}$ must be inside $convex\text{-}hull(S')$. The dotted region $R = MD(p_1, r') \cap MD(p_2, r')$ is the smallest TRR that contains S'. The hollow circles are the sinks in S' which lie on the boundary of R.

s can coincide at A). Furthermore, it is easy to see that $c(S')$ is the center of gravity of R. These facts imply that $c(S')$ lies inside *convex-hull*($\{x,y,s,t\}$). Since $\{x, y, s, t\} \subseteq S'$, we conclude that $c(S')$ lies inside *convex-hull*(S').

We now prove that the embedding and partitioning rules have the following properties. Our discussion again refers to Figure 8.

Theorem 2: Given a subset $S' \subseteq S$ with $|S'| \geq 2$, a convex polygon $P_{S'}$ containing S', and a point p inside $P_{S'}$, (i) the embedding rules will embed v inside $P_{S'}$ such that the embedding is compatible with the DME solution and (ii) the partitioning rules will divide $P_{S'}$ into two smaller convex polygons $P_{S_1'}$ and $P_{S_2'}$ that contain nonempty sink subsets S_1' and S_2', respectively, such that the routing from p to v is on the boundary between $P_{S_1'}$ and $P_{S_2'}$.

Proof: First, note that for all the embedding cases v lies on the portion of $center(S')$ that is closest to p; thus, v is a feasible DME embedding point.

Second, we show that v always lies inside $P_{S'}$, as follows. Recall that $\overline{AB}$ and $\overline{CD}$ must each contain a point of S'. Let x and y be these two points on $\overline{AB}$ and $\overline{CD}$ respectively. If point p lies inside Region I or Region III, then by Fact 4, $v = c(S')$ is inside *convex-hull*(S') which is contained in $P_{S'}$, whence v lies inside $P_{S'}$. If point p is in Region II or Region IV, it is easy to see from the Figure that v lies inside *convex-hull*$(\{p, c(S'), x, y\})$, which in turn lies inside *convex-hull*(S') and thus inside $P_{S'}$.

Third, we show that for each case for the location of p, $\overleftrightarrow{pv}$ divides S' into two non-empty subsets.

- $p \neq v$, and p located in Regions I, III: Since $v = c(S')$, by Fact 4 we have that v is inside *convex-hull*(S'), whence $\overleftrightarrow{pv}$ divides *convex-hull*(S') into two regions each containing at least one point from S'.
- $p \neq v$, and p located in Regions II, IV: The line $\overleftrightarrow{pv}$ separates $\overline{AB}$ from $\overline{CD}$.
- $v = p = c(S')$: The vertical line through v separates Regions I and III and hence separates $\overline{AB}$ and $\overline{CD}$.
- $v = p$, $p \neq c(S')$: The line $\overleftrightarrow{p_1p_2}$ again separates $\overline{AB}$ and $\overline{CD}$. □

Finally, inductive application of Theorem 2 easily yields:

Theorem 3: The zero-skew clock routing tree constructed by Linear-Planar-DME is planar.

Proof: Initially, there is a convex polygon P_S (e.g., a rectangle) which contains the set of sinks S and a clock location clk; the clock location clk is the embedding point of the parent node p of v, where v is the root of the ZST $T(S)$ in any DME solution.[4] The embedding rules guarantee that we can find embedding point $l(v)$ within P_S so that the routing from $l(p)$ to $l(v)$ lies within P_S. The partitioning rules guarantee that we can partition P into two smaller convex polygons P_{S_1} and P_{S_2} that respectively contain non-empty sink subsets S_1 and S_2, such that the routing from $l(p)$ to $l(v)$ is on the boundary between P_{S_1} and P_{S_2}. Node v will become the parent node for the ZSTs $T(S_1)$ and $T(S_2)$, and is contained in both P_{S_1} and P_{S_2}. Inductively, all future routing over

[4] If clk is not given, then we arbitrarily set $clk = c(S)$.

S_1 and S_2 will be confined within P_{S_1} and P_{S_2}, respectively. We conclude that no routing crosses any other. $\square$

3 The Elmore-Planar-DME Algorithm

Several works on clock tree design use the Elmore delay model, which is more accurate than linear delay [4, 5, 6, 12]. In this section, we sketch a simple method which is the first to achieve a single-layer Elmore-ZST, i.e., a ZST under the Elmore delay model. Note that under the Elmore delay model, the DME algorithm is no longer optimal: it does not necessarily return a minimum-cost ZST for given S and G [3, 5].[5] Also, the merging segment for the root of the subtree over $S' \subseteq S$ in the DME solution is no longer independent of the subtree connection topology. Hence, the bottom-up DME phase cannot be eliminated, i.e., Single-Pass DME cannot be applied to the Elmore delay model. To construct a low-cost planar-embedded Elmore-ZST, we propose the following Elmore-Planar-DME heuristic. Two important issues are: (i) choice of the topology G, and (ii) embedding to achieve zero Elmore delay skew.

First, any connection topology can be trivially embedded with exact zero skew onto a single routing layer; however, re-embedding the topology of a non-planar ZST (e.g., from [12]) onto a single layer can drastically increase the tree cost. The partial correspondence between linear delay and Elmore delay (at least in some technology regimes) suggests that the (optimum) Linear-Planar-DME solution can be re-embedded to have zero Elmore delay skew with very little increase in tree cost. Thus, Linear-Planar-DME is a natural choice for generating the connection topology within our approach.[6]

Second, given a Linear-Planar-DME solution, it is simple to obtain a planar Elmore-ZST by elongating tree edges in a bottom-up fashion to balance differences in sink delays (e.g., by the "snaking" method of [23]). In the experimental comparisons of Section 5 below, we call such an approach "Naive-Elmore-Planar-DME". We find that unneeded elongation of tree edges can be saved by iterating both the application of DME to the given topology and the bottom-up modification of any resulting non-planar routing, based on a "principle of least commitment". Planarity is enforced in bottom-up order, with planar-embedded subtrees being retained so that

[5] Let T_v denote the subtree rooted at node v in a zero-skew routing. Let $C(v)$ and $t_{ED}(v)$ respectively denote the total capacitance of T_v and the Elmore delay from v to each sink in T_v. Assume that loading capacitance $C(s_i)$ is given for each sink s_i. Finally, let r and c be the per unit wire resistance and capacitance, and let l_1 and l_2 be lengths of edges from v to v_1 and v_2, respectively. Then, C_v and $t_{ED}(v)$ for an internal node v with children v_1 and v_2 are calculated as follows[13, 21, 22]: $C(v) = C(v_1) + C(v_2) + c \cdot (l_1 + l_2)$, $t_{ED}(v) = t_{ED}(v_1) + r \cdot l_1 \cdot (\frac{c \cdot l_1}{2} + C(v_1))$ $= t_{ED}(v_2) + r \cdot l_2 \cdot (\frac{c \cdot l_2}{2} + C(v_2))$. Typically, we assume $t_{ED}(s_i) = 0$, i.e., there is no "internal delay" at a sink node.

[6] Interestingly, we find that relaxing the planar-embeddable constraint in variations of Linear-Planar-DME leads to improved planar Elmore-ZST solutions (see Section 4 below). This is possible because the method we use to achieve exact zero Elmore delay skew does not depend on an initial planar-embedded solution.

they remain planar, and routing at higher levels being modified. Whenever any non-planar routing at some level of the ZST is changed, the merging tree for the ZST above this level is rebuilt, and top-down DME embedding is applied to the new merging tree. The complementary processes of merging tree reconstruction and top-down embedding are iterated until the entire ZST is planar.

Again, we emphasize that the DME algorithm cannot guarantee optimal tree cost under the Elmore model. Thus, our approach only heuristically minimizes the cost of the output planar Elmore-ZST.

3.1 High-Level Description

Our method marks each point $v \in T$ as either planar or non-planar. An edge in T is a *planar edge* if both its endpoints are marked as planar. A path $s \rightsquigarrow t$ is a sequence of line segments from s to t; a *planar path* is a path that does not cross any planar edge of T. We use $cost(s \rightsquigarrow t)$ and $hops(s \rightsquigarrow t)$ to respectively denote the pathlength of a path and the number of segments in the path. Finally, the bounding box $bbox(s, t)$ denotes the smallest rectangle containing points s and t.

The Elmore-Planar-DME algorithm is described in Figure 9. For simplicity, the template assumes that no clock source location has been prescribed. Accommodating a fixed clock source is straightforward, as seen from Figure 6. Initially, a ZST T is obtained by applying the original DME algorithm (using the Elmore delay model) to the given topology G and sink set S. Then, every sink is marked planar and all other nodes are marked non-planar. As long as the ZST T has a non-planar node, Elmore-Planar-DME iterates at Steps 6 and 7. Note that Step 6 constructs the merging tree TS only for non-planar nodes in the upper part of the ZST; Step 7 calls Find_Exact_Placements(TS) in Figure 3 to embed the shrinking set of non-planar nodes.

Because non-planar nodes are made planar in bottom-up order, Procedure Rebuild-Tree-of-Segments identifies the lowest non-planar nodes in the tree, i.e., the node set A at level L of the tree. Nodes in A have planar children and will be made planar in the current iteration. Even though there may be other non-planar nodes whose children are all planar, their processing is deferred since subtrees at lower levels of the ZST tend to contain shorter tree edges, and it is easier for longer edges to detour around shorter edges than vice-versa. This same intuition suggests processing the nodes of A in order of increasing merging cost.

To make the discussion more concrete, for each non-planar node $v \in A$, let v have DME embedding point w and children s and t. If edges $\overline{sw}$ and $\overline{tw}$ do not cross any existing planar edges of T (i.e., edges in E), then v is marked planar (Step 6), and edges $\overline{sw}$ and $\overline{tw}$ are added into the set of planar edges E (Step 7). Otherwise, the non-planar routing at node v will be modified at Steps 9-12 as described below. The merging segment $ms(v)$ will be either reduced to v's current

Algorithm Elmore-Planar-DME (G, S)

Input: Topology G; set of sinks S

Output: Planar ZST T having topology G

1. ZST $T \leftarrow$ DME(G,S)
2. Mark all sinks of T planar
3. Mark all internal nodes of T non-planar
4. $E \leftarrow \emptyset$ /* set of planar edges */
5. **while** T still has a non-planar node **do**
6.   Merging tree $TS \leftarrow$ Rebuild-Tree-of-Segments(T, E)
7    $T \leftarrow$ Find_Exact_Placements(TS)

Procedure Rebuild-Tree-of-Segments(T, E)

Input: ZST T; Set of planar edges E

Output: Merging tree TS

1. $L \leftarrow$ Lowest level in T containing non-planar nodes
2. $A \leftarrow \{ v \mid v$ is non-planar and at level L in $T\}$
3. **for** each node $v \in A$ (increasing order of merging cost)
4.   Let node v have embedding point w and children s, t
5.   **if** $\overline{sw}$ and $\overline{tw}$ do not cross any planar edges
6.   Mark v as planar ($ms(v) \leftarrow \{l(v)\}$)
7. $E \leftarrow E \cup \{\overline{sw}, \overline{tw}\}$
8.   **else** /* modify non-planar routing at node v */
9.   Planar path $s \rightsquigarrow t \leftarrow$ Find-Merging-Path(E, v)
10.   **if** $cost(s \rightsquigarrow t) = d(s,t)$
11.   Improve-Path(E, v, $s \rightsquigarrow t$)
12.   Partial-Route(E, v, $s \rightsquigarrow t$)
13. Construct merging tree TS for all non-planar nodes

Figure 9: The Elmore-Planar-DME Algorithm.

embedding point if v is marked planar, or re-calculated if the non-planar routing at v is modified (see the discussion of subroutine Partial-Route below). Because the structure of the merging tree above the current level L will be changed, Step 13 constructs the tree of merging segments for the remaining non-planar nodes.

3.2 Modification of Non-Planar Routing

Now we consider the case where $\overline{sw}$ or $\overline{tw}$ crosses a planar edge. Recall that the DME embedding point w is the point on $ms(v)$ which is closest to the embedding point of v's sibling (so that the merging cost for node v and v's parent can be minimized). Our heuristic (Steps 9-12 of Rebuild-Tree-of-Segments) is to find a planar *merging path* $s \rightsquigarrow t$ such that $s \rightsquigarrow t$ is as short as possible and as near point w as possible. Specifically, we first use Procedure Find-Merging-Path to seek a planar *detouring path* $s \rightsquigarrow t$ with low merging cost at both v and p (e.g., see Figure 10). If the $s \rightsquigarrow t$ path has minimum possible pathlength ($= d(s,t)$), then Procedure Improve-Path is applied to further reduce the merging cost at v's parent by modifying the $s \rightsquigarrow t$ path so that it passes closer to the DME embedding point w without increasing its pathlength (e.g., see Figure 12). Otherwise, Procedure Partial-Route is used to bring s and t one hop closer together.

Details of the Subroutines

Details of **Procedure Find-Merging-Path** are given in Figure 11. We use the term *detour point* to denote an embedding point of a planar node which serves as an intermediate point in the $s \rightsquigarrow t$ path. Note that finding a shortest path over all detour points may not minimize the merging cost at p, and that slightly greater merging cost at v may result in much lower merging cost at p. Figure 10 shows an example in which path P_1 is slightly longer than path P_2, but is a better choice since it passes much closer to the DME embedding point w. To balance between efficiency and solution quality, Find-Merging-Path gradually increases the set of possible detour points, in the hope that a feasible path will be found early (i.e., when the problem size is small).

Let T_v denote the subtree of a ZST T rooted at point $v \in T$. Also recall that edge e_v denotes the edge connecting v and v's parent. Experimental results below use $V_1 = \{u \mid u \in T_v$, where edge e_v intersects $\overline{sw}$ or $\overline{tw}\}$ and $V_2 = \{ u \mid u \in T_v$, where edge e_v intersects $\overline{sw}$, $\overline{tw}$, or $\overline{st}\}$. For the example in Figure 10, Find-Merging-Path will use $V_1 = \{a, b, c\}$ and $V_2 = \{a, b, c, d, e, f\}$. These choices of V_1 and V_2 allow planar paths near w to be selected first. If Find-Merging-Path fails to discover a feasible path using V_1 and V_2, the procedure considers a succession of larger point sets V_i, $i \geq 3$; in our experiments, these are simply increasing dilations[7] of the bounding box $bbox(s, t)$.

Procedure Improve-Path in Figure 13 is applied only when the merging path $s \rightsquigarrow t$ obtained by Procedure Find-Merging-Path has minimum length equal to $d(s, t)$. The procedure tries to modify $s \rightsquigarrow t$ without increasing its length so that it passes closer to v's DME embedding point w. The procedure first selects a set of candidate embedding points on $ms(v)$. Then, each selected point u in increasing order of $d(u, w)$ is checked to see whether the shortest planar path $s \rightsquigarrow u \rightsquigarrow t$ has cost $= d(s, t)$. The shortest planar path $s \rightsquigarrow u \rightsquigarrow t$ is obtained by calling Find-Shortest-Planar-Path twice, i.e, by finding $s \rightsquigarrow u$ and $u \rightsquigarrow t$. Note that to find a minimum-cost path, say, $s \rightsquigarrow u$, we need only consider detour points inside $bbox(s, u)$. The procedure terminates when the first $s \rightsquigarrow u \rightsquigarrow t$ path with cost $= d(s, t)$ is found.

In addition to the intersection of $ms(v)$ and $s \rightsquigarrow t$ (shown as point u' in Figure 12), there are two types of candidate embedding points on $ms(v)$: (i) the intersection of $ms(v)$ with any vertical or horizontal line through any detour point inside $bbox(s, t)$ (see Figure 12a), and (ii) the intersection of $ms(v)$ with any planar edge (see Figure 12b). Again, the key property of u is that it is the point on $ms(v)$ closest to the DME embedding point w, such that the merging path through u still has minimum cost equal to $d(s, t)$.

[7] Our experiments increase each bounding box dimension by 10% at each iteration.

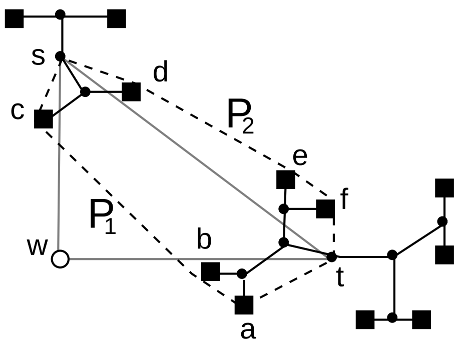


Figure 10: Find-Merging-Path heuristically searches for a planar merging path $s \rightsquigarrow t$ with low merging cost at both v and its parent p. Point w is the embedding point for node v that is computed by DME.

Procedure Find-Merging-Path(E, v)
Input: Set of planar edges E; Node $v \in G$ with children s and t
Output: Planar path $s \rightsquigarrow t$
$i \leftarrow 1$ **do** Construct $V_i \equiv i^{th}$ set of candidate detour points $s \rightsquigarrow t \leftarrow$ Find-Shortest-Planar-Path(E,V_i,s,t) $i \leftarrow i+1$ **while** ($s \rightsquigarrow t$ has not yet been found)

Figure 11: Procedure: Find-Merging-Path.

Procedure Partial-Route in Figure 14 uses a "principle of least commitment" whereby the distance between the two children of node v is shortened by one hop at each iteration. Suppose that the current non-planar node v has children s and t, and that we have a planar path $s \rightsquigarrow t = \{s, s', \cdots, u, \cdots, t', t\}$, with u being the point where zero skew is achieved. Without loss of generality, assume that $0 < d(s,s') \leq d(t,t')$. Then, Partial-Route implements only the partial path $\overline{ss'}$, with s' replacing s as a child of v and $\overline{ss'}$ being added to planar edge set E. In this way, v's children are "pulled closer" toward the delay balance point u so that v can be better re-embedded by DME in the next iteration. This avoidance of "commitment" also allows Partial-Route to minimize the harmful effects of a suboptimal result from Find-Merging-Path or Improve-Path.[8] Notice that since one of v's children is relocated, the merging segments for v and v's ancestors have to be re-calculated.

Finally, note that both Find-Merging-Path and Improve-Path invoke the procedure Find-Shortest-Planar-Path, which determines a shortest path between two points s and t in the presence of obstacles (the obstacles are the planar edges in subtrees of the ZST T). Since the detour points

[8]Thus, Procedure Rebuild-Tree-of-Segments may iterate several times at each level. In our experience, no more than 56 iterations in total were necessary for any of the benchmarks tested.

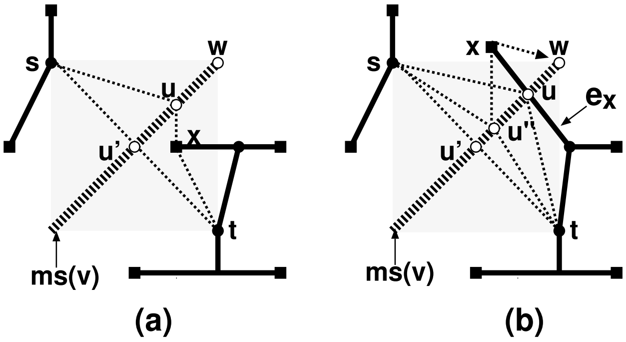


Figure 12: Improve-Path improves the planar merging path $s \rightsquigarrow t$ so that it passes closer to the DME embedding point w while retaining minimum merging cost $= d(s,t)$ for node v. Initially, path $s \rightsquigarrow u' \rightsquigarrow t$ is obtained by Procedure Find-Merging-Path, and is then improved to $s \rightsquigarrow u \rightsquigarrow t$ if $cost(s \rightsquigarrow u' \rightsquigarrow t) = d(s,t)$. In (a), $u = ms(v) \cap$ the vertical line through a detour point x inside $bbox(s,t)$ (the dotted region). In (b), $u = ms(v) \cap e_x$, and $u'' = ms(v) \cap$ {the vertical line through a detour point x outside $bbox(s,t)$}. Both paths $s \rightsquigarrow u \rightsquigarrow t$ and $s \rightsquigarrow u'' \rightsquigarrow t$ will actually yield the same merging cost savings at v's parent. For such purposes as crosstalk minimization, u'' may actually be a better choice than u. However, in our present implementation we simply select u.

Procedure Improve-Path(E,v, $s \rightsquigarrow t$)

Input: Set of planar edges E; Non-planar node v with children s, t and DME embedding point w; planar path $s \rightsquigarrow t$ with $cost(s \rightsquigarrow t) = d(s,t)$

Output: Planar path $s \rightsquigarrow u \rightsquigarrow t$ s.t. $cost(s \rightsquigarrow u \rightsquigarrow t) = d(s,t)$ and $d(u,w)$ are minimal for point $u \in ms(v)$.

$A = \{ u \mid u = ms(v) \cap l$, where l is a horizontal or vertical line through a detour point in $bbox(s,t)\}$
$B = \{ u \mid u = ms(v) \cap l$, where l is a planar edge in $E\}$
$C = \{ u \mid u = ms(v) \cap s \rightsquigarrow t\}$
Construct $U = A \cup B \cup C$
do
  Select point $u \in U$ in increasing order of $d(u,w)$
  $D =$ set of detour points inside $bbox(s,u)$
  $s \rightsquigarrow u =$ Find-Shortest-Planar-Path(E,D,s,u)
  $D =$ set of detour points inside $bbox(t,u)$
  $u \rightsquigarrow t =$ Find-Shortest-Planar-Path(E,D,u,t)
  $s \rightsquigarrow t = s \rightsquigarrow u \cup u \rightsquigarrow t$
while $cost(s \rightsquigarrow t) > d(s,t)$

Figure 13: Procedure Improve-Path.

must be located at the endpoints of planar edges, a general approach based on visibility graphs (e.g., [1, 24]) can be used. Our current implementation uses Dijkstra's algorithm in the visibility graph, with edge weights computed on the fly; this does not cause excessive runtimes (see Section 5) since the number of possible detour points is small in most procedure calls.

Procedure Partial-Route(E, v,$s \rightsquigarrow t$)

Input: Set of planar edges E;
Non-planar node v with children s and t;
Merging path $s \rightsquigarrow t = \{s, s', \cdots, u, \cdots, t', t\}$,
where $u =$ the delay balance point.

Output: Partial implementation of $s \rightsquigarrow t$;
New location for one of v's children;
Updated planar edge set E

if $0 < d(s \rightsquigarrow s') \leq d(t, t')$
  Connect a wire from s' to s
  $s = s'$;
  $E \leftarrow E \cup \{\overline{ss'}\}$
else
  Connect a wire from t' to t
  $t = t'$;
  $E \leftarrow E \cup \{\overline{tt'}\}$

Figure 14: Procedure Partial-Route.

4 Linear-Planar-DME Variants

As noted above, Elmore-Planar-DME does not actually require a planar-embedded ZST as input. Thus, while we use the topology G obtained from the Linear-Planar-DME solution, the Linear-Planar-DME construction itself can actually be modified.

We have considered modifications to the partitioning rules of Section 2.2 which change the splitting line to a splitting *path* of two or more line segments. In other words, rather than draw a straight line through points p and v, we draw a line segment $\overline{pv}$ and a ray $\vec{v}$ emanating from v to separate the polygons P_{S_1} and P_{S_2}. Since we no longer have a straight splitting line, one of the new smaller regions may be non-convex, and more case analysis is required to maintain guaranteed planarity of the output ZST. From a theoretical perspective, such Linear-Planar-DME variants are unappealing: we lose the guaranteed planarity, and the worst-case time complexity increases. However, all ZST's obtained in our experiments remain planar, with non-convex polygons becoming further divided into smaller convex polygons within the succeeding two or three levels. Furthermore, such variants can achieve averages of up to 10.9% wirelength reduction versus results for the original Linear-Planar-DME algorithm which we have reported in [17]. We now briefly describe two possible Linear-Planar-DME variants.

Using a Splitting Path

Consider a subset of sinks $S' \subseteq S$ that is being partitioned, with $|S'| \geq 2$. Recall that the splitting path consists of line segment $\overline{pv}$ and $\vec{v}$, a ray emanating from v. The line segment $\overline{pv}$ has been determined, but there are $|S'| - 1$ different choices of $\vec{v}$. To consider all possible choices of $\vec{v}$, our

Linear-Planar-DME-2 variant sorts the sinks of S' in clockwise order around point v; each pair of consecutive sinks determines a splitting path which partitions S' into S'_1 and S'_2. To choose among the possible splitting paths, Linear-Planar-DME-2 uses a heuristic estimate of the cost of the ZSTs over S'_1 and S'_2. We have experimentally determined two such estimates:

- $r_1 \times |S'_1| + r_2 \times |S'_2|$, where r_1 and r_2 are the respective radii of the sink sets S'_1 and S'_2; and
- $r_1 \times |S'_1| + r_2 \times |S'_2| + 0.5r(|c_1 - c_2|/c)^2$, where r is the radius of S' and c, c_1 and c_2 are the respective total capacitances of the sink sets S', S'_1 and S'_2.

The latter estimate considers load balance when bipartitioning the sinks, and yields slightly better results (it is also the estimate used in the experiments reported below). More useful cost functions for sink partitioning are no doubt possible.

In the Manhattan plane, computing the radii of all pairs of sink subsets (corresponding to bipartitions of S') can be accomplished in $O(|S'|)$ time. Thus, the sorting operation dominates the time complexity, and the overall Linear-Planar-DME-2 complexity is $O(l \cdot n \lg n)$, where l is the number of levels in the output ZST and $n = |S|$. In practice, l is very close to $\lg n$, as we report below.

Using a Splitting Path and Lookahead

Our Linear-Planar-DME-3 variant is similar to Linear-Planar-DME-2, but chooses splitting paths more carefully based on lookahead. After determining a set of candidate bipartitions of S', we estimate the cost of each by actually constructing the ZST that will be output by Linear-Planar-DME-2. To maintain practical runtimes, the number of candidate bipartitions considered is limited to a small constant (≈ 16 in the experiments reported below). Given this constraint, our Linear-Planar-DME-3 implementation has worst-case runtime of $O(l^2 \cdot n \lg n)$.

Finally, if the clock source is not specified, then the line segment $\overline{pv}$ of the first splitting line can be arbitrarily determined since p's location is not given. As we determine the possible choices of $\vec{v}$, we sort the sinks clockwise around v; each pair of consecutive sinks determines a possible choice of $\overline{pv}$. Thus, there are $|S|$ possible cases for p's location. Again, to maintain practical runtime, we test only 16 equally spaced cases for p's locations. Experimentally, very limited improvements result from trying more than 16 cases.

5 Experimental Results

We implemented the Linear-Planar-DME and Elmore-Planar-DME algorithms using Sun SPARC-10 workstations and the C/Unix environment. The same seven examples as in [5, 12, 25] were studied. Benchmarks Primary1 and Primary2 both have the same loading capacitance of $0.5pF$ for all sinks, and also have per-unit wire resistance and wire capacitance of $16.6m\Omega$ and $0.027fF$, respectively. The x-coordinates and y-coordinates of Primary1 sink locations range from 120 to 5520 units and from 0 to 5790 units, respectively; those of Primary2 range from 20 to 9840 units and from 0 to 10250 units, respectively. Details of the circuit parameters for benchmarks r1-r5 can be found in [23].

Table 1 shows that our Elmore-Planar-DME implementation is relatively efficient, with runtimes dominated by the generation of a good topology in the call to Linear-Planar-DME-3. Note that the Primary2 and r2 test cases have about the same number of sinks, but Primary2 leads to relatively higher runtimes. This is because Primary2 has a more uneven distribution of sink locations, which leads to more detouring. The last column of Table 1 shows that our output ZSTs have very balanced structures, with average tree height l very close to $\lg n$. Thus, the observed time complexity of Linear-Planar-DME-3 is $O(n \cdot (\lg n)^3)$.

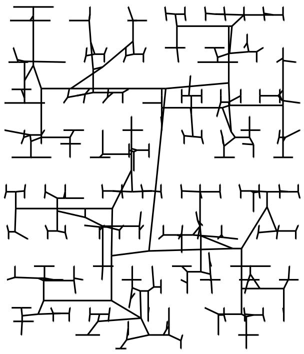
benchmark (#pins)	Lin-Pln-DME-3	Elm-Pln-DME	ZST height ($l/\lg n$)
prim1(269)	115	1	9 (1.1)
prim2(603)	362	34	11 (1.2)
r1(267)	114	4	11 (1.4)
r2(598)	382	13	12 (1.3)
r3(862)	664	36	12 (1.2)
r4(1,903)	2512	327	14 (1.3)
r5(3,101)	5557	339	15 (1.3)

Table 1: Sun SPARC-10 CPU time (seconds) for our Planar-DME implementation. Note that the topology generation via Linear-Planar-DME-3 requires much more time than the embedding by Elmore-Planar-DME. In the last column, we also show the ZST height as a multiple of the minimum possible tree height, $\lg n$.

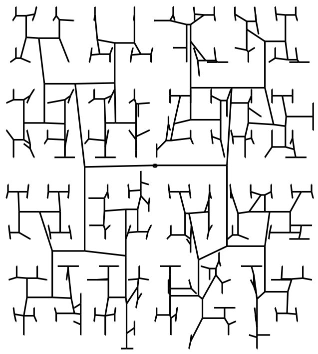
Table 2 compares our new algorithms with two leading non-planar ZST algorithms in the literature – Greedy-DME [12] and KCR+DME [5, 18] – as well as the previous planar routing method of Zhu and Dai [25]. Greedy-DME corresponds to the CL+I6 method of Edahiro [12], and can yield an unbalanced topology. KCR+DME uses a matching approach to achieve a balanced topology [5]. Our new planar ZST solutions are competitive with the best known non-planar ZST solutions of Greedy-DME (having average 9.8% greater wiring cost), and are superior to KCR+DME solutions in all cases. Elmore-Planar-DME also uses 22.5% less wire than the (linear delay based) method of [25]. It is interesting to note that the cost of our Elmore-Planar-DME solutions is only slightly increased from the cost of the starting Linear-Planar-DME ZSTs. We believe this implies that better solutions can be obtained as we continue to improve Linear-Planar-DME. Finally, Figure 15 shows ZSTs for the Primary1 benchmark constructed by Greedy-DME, Linear-Planar-DME, Elmore-Planar-DME, and the method of Zhu and Dai.

benchmark	Greedy-DME (CL+I6[12])	Lin-Pln-DME-3	Elm-Pln-DME	KCR+DME [5]	Naive Elm-Pln-DME	Zhu-Dai [25]
prim1	129.2	130.2	132.9	140.1	146.1	167.9
prim2	304.0	320.1	330.2	345.2	391.6	422.5
r1	1,253.3	1,351.3	1,392.8	1,487	1,686.2	1,778.3
r2	2,483.8	2,725.1	2,789.3	3,020	3,315.2	3,580.1
r3	3,193.8	3,501.4	3,543.7	3,867	3,916.2	4,635.9
r4	6,499.7	7,073.4	7,261.7	7,713	8,582.5	9,577.1
r5	9,723.7	10,556.2	10,806.4	11,606	12,823.3	14,119.4
Ave Cost	(+0.0%)	+7.2%	+9.8%	+17.3%	+28.1%	+39.4%
Planar	No	Yes	Yes	No	Yes	Yes
Delay Model	Elmore	Linear	Elmore	Elmore	Elmore	Linear

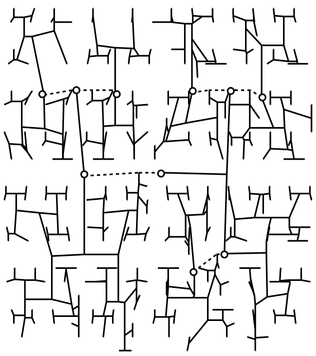
Table 2: Comparison of Elmore-Planar-DME with other algorithms in terms of total wirelength, using the same benchmarks studied in [5, 12, 25]. No prescribed clock source location was assumed. Ave Cost indicates the average percentage increase in wirelength versus the results of CL+I6 [12]. Note that all the wirelength has been divided by 1000 units.



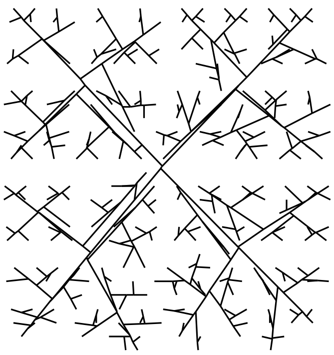
(a) Greedy-DME [12]



(b) Linear-Planar-DME



(c) Elmore-Planar-DME



(d) Zhu-Dai [25]

Figure 15: Zero-skew clock routing solutions for the Primary1 benchmark. Six instances of detour routing in (c) are highlighted with dotted lines.

6 Future Work

We have considered several improvements to our current work.

- First, the output of our Planar-DME approach may be viewed as a *planar routing sketch* for a ZST. Currently, we do not take routing capacity, cross-talk constraints, etc. into consideration (recall the example of Figure 12b). We hope to use such computational geometry techniques as those of Dai et al. [9] to enhance our current approach.
- Second, although Elmore-Planar-DME has reasonable runtime in practice, various heuristic speedups are possible. For example, obstacles (planar edges) are actually connected as subtrees, and each subtree can be replaced by its convex hull to reduce the complexity of the path-finding instance. Also the number of candidate embedding points tested by Improve-Path can be greatly reduced.
- Third, Linear-Planar-DME itself can be improved to yield better connection topologies for input to Elmore-Planar-DME, through the use of more sophisticated partitioning rules (using splitting paths with more than two segments; clustering sinks before partitioning) and embedding rules (e.g., embed the root of the zero-skew subtree over S' at a more appropriate place than $center(S')$).
- Finally, we are pursuing methods which construct single-layer clock routing trees with *bounded*, rather than exactly zero, skew; such constructions are useful in the engineering of general clock distribution solutions, where skew and other attributes are controlled by a mix of topology generation, embedding, wiresizing and buffer optimization [7, 14, 15, 20]

References

[1] T. Asano, T, Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility-polygon search and Euclidean shortest paths", *Proc. IEEE Symp. on Foundations of Computer Science*, 1985, pp. 155-164.

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

[5] T.-H. Chao, Y. C. Hsu, J. M. Ho, K. D. Boese and A. B. Kahng, "Zero Skew Clock Routing With Minimum Wirelength", *IEEE Trans. on Circuits and Systems* 39(11) (1992), pp. 799-814.

[6] N.-C. Chou and C.-.K. Cheng, "Wire Length and Delay Minimization in General Clock Net Routing", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1993, pp. 552-555.

[7] J. Cong, A. B. Kahng, C.-K. Koh and C.-W. A. Tsao, "Bounded-Skew Clock and Steiner Routing Under Elmore Delay", to appear in *Proc. IEEE Intl. Conf. on Computer Aided Design*, San Jose, November 5-9, 1995.

[8] J. Cong, A. B. Kahng and G. Robins, "Matching-Based Methods for High-Performance Clock Routing", *IEEE Trans. on CAD* 12(8), August 1993, pp. 1157-1169.

[9] W. M. Dai, R. Kong, J. Jue, and M. Sato, "Rubber band routing and dynamic data representation", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 52-55.

[10] D. Dobberpuhl et al., "A 200MHz 64b Dual-Issue CMOS Microprocessor", *Proc. IEEE Intl. Solid State Circuits Conf.*, Feb. 1992, pp. 106-107.

[11] M. Edahiro, "Minimum Skew and Minimum Path Length Routing in VLSI Layout Design", *NEC Research and Development* 32(4), October 1991, pp. 569-575.

[12] M. Edahiro, "Clustering-Based Optimization Algorithm in Zero-Skew Routings", *Proc. ACM/IEEE Design Automation Conf.*, June 1993, pp. 612-616.

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

[14] E. G. Friedman, "Clock Distribution Design in VLSI Circuits – An Overview", *Proc. IEEE ISCAS*, May 1993, pp. 1475-1478.

[15] J. H. Huang, A. B. Kahng and C.-W. A. Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", *Proc. 32nd ACM/IEEE Design Automation Conf.*, San Francisco, June 1995, pp. 508-513.

[16] 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.

[17] A. B. Kahng and C.-W. A. Tsao. "Planar-DME: Improved Planar Zero-Skew Clock Routing With Minimum Pathlength Delay," *Proc. ACM/IEEE European Design Automation Conf.*, September 1994.

[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] W. Khan, X. He, L. Bangaru and N. Sherwani, "Combat: Zero Skew Minimal Delay Planar Clock Routing for High Performance Systems", *Western Michigan Univ. Computer Science Technical Report TR/93-08*, April 15, 1993

[20] S. Pullela, N. Menezes, J. Omar and L. T. Pillage, "Skew and Delay Optimization for Reliable Buffered Clock Trees", *Proc. IEEE Intl. Conference on Computer-Aided Design*, Nov. 1993, pp. 556-562.

[21] 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.

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

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

[24] E. Welzl, "Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time", *Information Processing Letters* 20 (1985), pp. 167-171.

[25] Q. Zhu and W. W.-M. Dai, "Perfect-Balance Planar Clock Routing With Minimal Path-Length", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1992, pp. 473-476.