Timing Optimization for Multisource Nets: Characterization and Optimal Repeater Insertion

John Lillis and Chung-Kuan Cheng, *Senior Member, IEEE*

***Abstract*—This paper presents new results in the area of timing optimization for multisource nets. The augmented RC-diameter (ARD) is suggested as a natural and practical performance measure and a linear time algorithm for computing the ARD of a multisource net is presented. Building on the ARD measure, we characterize the multisource optimization problem in terms of operations on piece-wise linear functions. This characterization is then used to develop an algorithm for *optimal repeater insertion*: for a given multisource topology the algorithm efficiently identifies an optimal assignment of repeaters to prescribed insertion points under the "min cost timing feasible" problem formulation. The algorithm has been implemented and computational results demonstrate the viability of the approach.**

***Index Terms*—Buffering, dynamic programming, interconnect, optimization, timing.**

I. Introduction

RECENT years have seen much research in automatic timing optimization of interconnect for VLSI systems. Much of this work has been driven by a technological shift in the relative importance of interconnect delay and logic delay; due to increased wire resistance at ever shrinking geometries, interconnect now plays an increasingly dominant role in overall performance. Techniques for interconnect optimization include buffer insertion (e.g., [1], [9], [15], and [26]), performance driven topology synthesis (e.g., [2], [5], [11], [16], and [27]), wire sizing (e.g., [4], [15], and [22]) and various combinations of these techniques (e.g., [17], and [20]). We note that all of these cited works focus on optimizing *single source* routing topologies.

After considering the various single-source formulations, optimization techniques and algorithms, it is natural to ask what can be done in the case of multisource nets since buses are so prevalent in modern designs. This topic has only recently received attention (e.g., [6], [7], and [8], [24]) and is the topic of this paper. Our primary contributions in this area are as follows.

1) We adopt the augmented RC-diameter (ARD), a natural performance measure based on RC delay models, the arrival times at the inputs (sources) of a multisource, and downstream delays from the sinks (outputs) of the net. The ARD is a straightforward generalization of the typical single-source performance measures and is, we feel, more appropriate than measures such as weighted sums of pair-wise delays (e.g., [7], and [8]) which only have an indirect relation to system objectives such as clock period.
2) We demonstrate that the ARD of a given topology can be computed in $O(n)$ time and, thus, is no harder than computing an RC-radius (see, e.g., [18], [21], and [25])—i.e., it is unnecessary to perform multiple single-source computations.
3) Most significantly, we present an algorithm for optimal repeater (i.e., bidirectional buffer) insertion for a given routing topology.[1] We adopt the objective of cost (e.g., area) minimization subject to a given timing spec. As in some previous work (e.g., [16]), the technique produces a suite of solutions which exhibit a cost versus performance tradeoff. Finally, we note that the technique subsumes the discrete driver sizing problem (subject to the drivers effectively being "two-stage"). In the process we develop general techniques for characterizing multisource timing optimization using manipulation of piece-wise linear (PWL) functions. These fundamental operations have application beyond the particular problem of repeater insertion.

We focus on bidirectional repeater insertion in part because it has achieved a certain degree of maturity in the high-end design community and in fact is commonly used in the design of high-speed commercial microprocessors [19]. However, it was also selected because it exercises many of the issues involved in multisource optimization and thus serves as a good testbed in which to illustrate these techniques.[2]

We note that there are some important design and technology issues in repeater insertion which do not appear in the single source problem. The primary issue is the choice of repeater design; should one use a pair of tristate buffers controlled by a bus arbitrator[3] or more sophisticated "direction sensing" autonomous repeaters (e.g., [12], and [13])? Such decisions are largely dependent on designer preference [19]

Manuscript received March 27, 1998; revised July 1, 1998. This paper was recommended by Associate Editor M. Sarrafzadeh

J. Lillis is with the Department of Electrical Engineering and Computer Science, Univeristy of Illinois at Chicago, Chicago, IL 60607-7053 USA.

C.-K. Cheng is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA.

Publisher Item Identifier S 0278-0070(99)01588-2.

[1] Throughout the paper we use the term "repeater" to refer to a bidirectional buffer.

[2] We emphasize that the techniques presented herein are fairly general and are, conceptually, relatively easily adapted to a variety of multisource optimization techniques where a dynamic-programming framework is utilized (e.g., it is conceptually straight forward to incorporate these techniques into a topology construction procedure such as [16]).

[3] Note that a bus controller is typically necessary whether repeaters are inserted or not and thus the additional control overhead when repeaters are used can reasonably be expected to be small.

0278–0070/99$10.00 © 1999 IEEE

and are not addressed here. Fortunately, such issues appear to have little impact on the fundamental nature of the optimization problem and, thus, the techniques presented here remain relevant independent of such decisions.

To put our contributions into perspective, we briefly review the most relevant previous work in the area. A well known property of the Elmore model is that maximum source-to-sink delay (i.e., *RC-radius*) [10] can be computed in linear time by depth-first search [18], [21], [25] (in fact, *all* source-to sink delays can be computed, but the RC-radius is of primary interest here). The technique easily accommodates generalizations in which buffers may be inserted and sinks have specific timing requirements. Our second contribution above can be thought of as a generalization of the single source case; it also demonstrates that it is unnecessary to perform multiple single source computations in order to determine RC-diameter.

In [26], van Ginneken presented an elegant dynamic programming algorithm for optimal buffer insertion into a given single-source routing topology under a basic buffer delay model. For historical accuracy, we point out that Dhar independently discovered a very similar algorithm [9]. In [15], Lillis *et al*. gave a detailed and efficient implementation of the "min-cost subject to timing requirements" formulation of the single-source buffer insertion problem; the techniques included simultaneous wire sizing and a generalized buffer delay model incorporating signal slew. Additionally, some recent efforts have been made at simultaneous buffer insertion and topology synthesis [17], [20]. In [24], Tsai *et al*. proposed a heuristic solution to the repeater insertion problem in which timing requirements were assumed to be given as arbitrary pair-wise constraints; their algorithm was based on local optimization and convex programming. For two pin nets, closed forms for delay optimal buffer spacing have also been known for some time (see [1], for example).

Additional work in the multisource domain includes [6] in which Madden and Cong propose an algorithm for constructing low cost *minimum diameter* rectilinear Steiner trees ("diameter" being in the sense of geometric path length rather than RC delay). In [7], Cong and He studied the problem of minimizing a *weighted sum* of pair-wise Elmore delays by wire sizing. Additionally, the convex programming techniques of [22] for *continuous* wire sizing can be generalized to accommodate the multisource case including driver sizing [23].

The remainder of this paper is organized as follows. Section II introduces nomenclature, delay models, and problem formulations. Section III presents an algorithm for computing the augmented RC-diameter of a multisource topology. Section IV presents an algorithm for optimal repeater insertion; the algorithm is based on functional abstraction and dynamic programming. Section V presents preliminary experimental results and we conclude in Section VI.

II. Preliminaries and Formulations

Throughout the remainder of this paper, the following are assumed to be given technology parameters.

- α and β which are, respectively, resistance (in ohms) and capacitance (in pF) per unit wire length (e.g., 1 μm) in the target technology.
- A library L_b of repeaters. Each repeater has an "*A-side*" and a "*B-side*" allowing us to refer to signal flow as *A-to-B* or *B-to-A* and thereby take into account the orientation of the repeater. The following parameters characterize a repeater where subscripts indicate signal direction:

$i_{AB}(b)$, $i_{BA}(b)$	intrinsic delay of b;
$r_{AB}(b)$, $r_{BA}(b)$	output resistance of b;
$c_{AB}(b)$, $c_{BA}(b)$	input capacitance of b;
$s(b)$	cost of b (e.g., area).

For completeness, we review the Elmore model for wire delay. The delay of a signal transmitted along a wire of length l from u to v is given as

$$\alpha l\left(\frac{l\beta}{2}+c_v\right)$$

where c_v is the lumped downstream capacitance at v.[4] Similarly, the typical basic model the delay of a buffer b is the sum of b's intrinsic delay and RC delay given as

$$i(b)+r(b)\cdot c_{\text{load}}$$

where c_{load} is the capacitive load on the output of b. Naturally, in the case of a repeater, this delay depends on the signal direction and the parameters are used accordingly. Subsequently when referring to the delay of a path, we assume these models are applied.

In addition to the above technology parameters, the following net-specific parameters are also given.

- Terminal set N in the plane; each terminal may act as an input or output and thus may have two buffers—an input buffer which drives the bus and an output buffer used when the terminal acts as a sink. The following parameters are associated with each terminal $v \in N$.
  - $a(v)$ Maximum delay from a primary input (PI) of the circuit to the input buffer at v.
  - $d(v)$ Maximum delay from output buffer at v to a primary output (PO) of the circuit.[5]
  - $c(v)$ Input capacitance associated with input buffer at v.
  - $r(v)$ Output resistance of the input buffer at v when acting as a source.
- A Rectilinear Steiner Tree T connecting N with prescribed, degree two, candidate buffer insertion points.[6]

The parameters associated with a terminal are illustrated in Fig. 1. In the figure the terminal v serves as both a source and a sink by using the driver component enclosed in dashed lines. When acting as a source, its input arrives at time $a(v)$;

[4] Note that for fixed width wires as discussed here, fringe capacitance can easily be incorporated.

[5] In this case the intrinsic and RC delay of the output buffer *it is* included since it is independent of the bus.

[6] We assume that insertion points have degree two to avoid ambiguity with respect to which side of the repeater a branch connects.

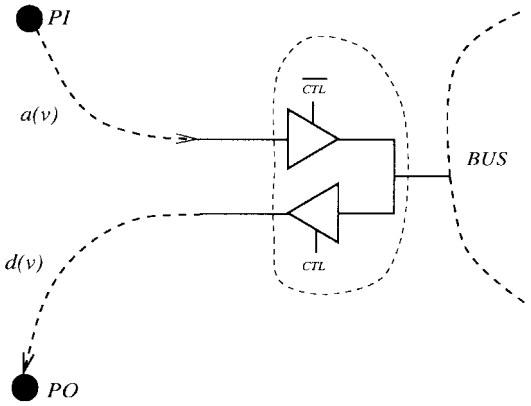


Fig. 1. Parameters associated with a terminal v.

conversely, when acting as a sink, the signal it produces will experience $d(v)$ additional delay to a PO of the circuit.

To assess the performance of a routing tree T spanning terminal set N with a driver assignment and a (perhaps null) buffer assignment to the candidate insertion points we now formally define the augmented RC-diameter, ARD(T).

Definition 2.1: Let $PD(u, v)$ be the RC path delay from source pin u to sink pin v in topology T spanning N including delay of the driver (if it exists), buffers and wires on the path. We define the augmented RC diameter of T, ARD(T) as

$$\text{ARD(T)} = \max_{u \in N} \max_{v \in N/\{u\}} (a(u) + PD(u, v) + d(v)).$$

The ARD(T) captures the maximum delay from the circuit's PI's to its PO's among all paths traversing net N and thus, captures the impact of the solution on, for example, system cycle time. In the next section, we present a linear time algorithm for computing ARD(T) where Elmore is used in the delay calculation.[7]

The definition of ARD(T) leads naturally to the *optimal repeater insertion problem* as follows.

Problem 2.1: Given routing topology T spanning multi-source terminal set N, a set of technology parameters and performance spec D_{spec}, find an assignment and orientation of repeaters to the insertion points of T such that the resulting cost is minimized subject to ARD(T) $\leq D_{\text{spec}}$.

We present a solution to Problem 2.1 in Section IV.

We note that alternative formulations of the problem are possible. For instance, one may wish to minimize ARD(T) without regard to solution cost. However, we propose that Problem 2.1 is of more practical value (and in fact subsumes the cost oblivious formulation). One may also consider a formulation in which arbitrary delay constraints are given for each source/sink pair. While Problem 2.1 implicitly imposes pair-wise delay bounds, these bounds are not arbitrary as they are derived from the linear number of given parameters. We do not study the arbitrary pair-wise constrained problem here but we note that this problem appears significantly more complex; fortunately, Problem 2.1 appears to capture the typical scenario in practice. Finally, we point out that no generality is lost by not explicitly specifying the sources and sinks of the net; for any nonsink v, we can simply set $d(v) = -\infty$ and for any nonsource u, we can set $a(v) = -\infty$.

[7]Note, however that the concept of the ARD does not rely on the Elmore delay model; indeed the ARD is well defined regardless of how $PD(u, v)$ is calculated.

III. Linear Time Computation of ARD(T) Under Elmore

In the multisource formulation presented in the preceding section, the performance metric ARD(T) is expressed as a maximum over a set of single-source delay computations. Clearly, by performing n single-source computations we can compute ARD(T) in $O(n^2)$ time. However, we will show that ARD(T)—under Elmore—can be computed in linear time and thus is no harder (asymptotically) than computing the analogous performance metric for a single source net (e.g., maximum source-to-sink delay).[8]

Before proceeding, we clarify some additional assumptions and notation. First, without loss of generality, we assume that all terminals of the net are also leaves in the topology T; note that any nonleaf terminal can be made a leaf by adding a new vertex and a zero-length edge. For purposes of depth-first traversal, it will be useful to orient a topology T with respect to an (arbitrary) root vertex s. In such a rooted tree, we use $p(v)$ to indicate the parent of vertex v; we also note that edges $(u, v) \in T$ are now directed. We use $c(u, v)$ and $r(u, v)$ to, respectively, denote the capacitance and resistance of a wire $(u, v) \in T$. Finally, recall that we assume that repeaters are inserted along wires rather than at branch points.

The first step in computing ARD(T) in linear time will be the computation of the capacitive values $c_L(u, v)$ and $c_L(v, u)$ for every edge $(u, v) \in T$ where $c_L(u, v)$ is the total downstream capacitance at v *as seen from* u and likewise, $c_L(v, u)$ is the load at u from v's perspective. Both of these load values are needed because a signal may traverse edge (u, v) in either direction. We first compute $c_L(u, v)$ where $(u, v) \in T$ by the following recurrence[9]

$$c_L(u, v) = \begin{cases} c(v), & \text{if } v \text{ is a terminal} \\ c_{AB}(b), & \text{else if buffer } b \text{ placed at } v \\ \displaystyle\sum_{(v, w) \in T} c_L(v, w) + c(v, w), & \text{o.w.} \end{cases} \tag{1}$$

After this bottom-up process, we compute $c_L(v, u)$ where $(u, v) \in T$ top-down as follows.

$$c_L(v, u) = \begin{cases} c(u), & \text{if } u \text{ is the root} \\ c_{BA}(b), & \text{else if buffer } b \text{ placed at } u \\ c_L(u, p(u)) + c(u, p(u)) & \\ \quad + \displaystyle\sum_{\substack{(u, w) \in T \\ w \neq v}} c_L(u, w) + c(u, w), & \text{o.w.} \end{cases} \tag{2}$$

[8]This result gives some idea of the distinction between the formulation we propose and the "arbitrary pair-wise constraint" formulation where an $\Omega(n^2)$ time is necessary simply to examine all constraints.

[9]This recurrence and the subsequent recurrence assume without loss of generality that the A-side of buffer b connects to the parent in the re-oriented tree; simply reversing the subscripts of $c(b)$ accounts for the opposite orientation.

Routine: ARD(T)
Input: Multi-source topology T spanning terminal set N
Output: The Augmented RC-Diameter of T

1. Orient T with an arbitrary pin $s \in N$ as the root
2. Compute $c_L(u,v)$ and $c_L(v,u)$ for all $(u,v) \in T$
3. $(a, d, d_I) \leftarrow$ ARD_R(s) /* invoke recursive routine */
4. Let $c_L(s)$ be the total load on pin s
5. RETURN $\max(a + d(s), d + a(s) + r(s)c_L(s), d_I)$

Routine: ARD_R(v)
Input: vertex $v \in T$
Output: triple (a, d, d_I) where

$$a = \max_{u \in T_v \cap N} a(u) + PD(u,v)$$
$$d = \max_{u \in T_v \cap N} PD(v,u) + d(u)$$
$$d_I = \max_{u \in T_v \cap N} \max_{\substack{w \in T_v \cap N \\ w \neq u}} a(u) + PD(u,w) + d(w)$$

IF(v is a leaf)
  RETURN $(a(v) + r(v)(c_L(v,p(v)) + c(v,p(v))), d(v), -\infty)$
ELSE
  $a \leftarrow d \leftarrow d_I \leftarrow -\infty$
  FOREACH child u of v
    $(a(u), d(u), d_I(u)) \leftarrow$ ARD_R(u)
    $a'(u) \leftarrow a(u) + r(u,v)(\frac{c(u,v)}{2} + c_L(u,v))$
    $d'(u) \leftarrow d(u) + r(v,u)(\frac{c(v,u)}{2} + c_L(v,u))$
    IF (buffer b placed at v)
      $a'(u) \leftarrow a'(u) + r_{AB}(b)(c_L(v,p(v)) + c(v,p(u)))$ /* or use c_{BA} depending on orientation */
      $d'(u) \leftarrow d'(u) + r_{BA}(b)(c_L(v,u) + c(v,u))$ /* or use r_{AB} depending on orientation */
    $d_I \leftarrow \max(d_I, a + d'(u), d + a'(u), d_I(u))$
    $a \leftarrow \max(a, a'(u))$
    $d \leftarrow \max(d, d'(u))$
  RETURN (a, d, d_I)

Fig. 2. Pseudocode for computing ARD(T).

Finally, the overall algorithm for computing ARD(T) appears in Fig. 2. The algorithm recursively computes three values for each subtree T_v: a is the maximum augmented arrival time at v via sources in T_v; d is the maximum augmented delay from v to sinks in T_v; d_I is the maximum augmented RC-diameter among source/sink pairs in T_v. Once the root is reached, we can derive the ARD of the entire tree.

The algorithm just presented applies to the Elmore delay measure. However, as previously stated, the ARD is well-defined for any delay measure. In any such delay measure, if the delay along wire segments are known, the resulting ARD can easily be computed in linear time also by using depth-first search. Details are left to the reader.

IV. The Repeater Insertion Algorithm

As in the previous section, assume that the routing topology T has been reoriented with an arbitrary terminal designated as the root. Our algorithm for optimal repeater insertion is based on bottom-up dynamic programming on this reoriented tree.

A. Motivational Example

As in any dynamic programming approach, it is necessary to characterize the salient features of a subsolution and how they may affect a global solution. Consider a candidate repeater assignment s to a subtree rooted at vertex v; we refer to such a subtree as T_v. Intuitively, we must capture the following characteristics of the assignment:

1) cost;
2) capacitance;
3) maximum augmented delay from v to a sink in T_v [i.e., $\max_{u \in N \cap T_v} (PD(v, u) + d(u))$];
4) maximum arrival time at v from sources in T_v;
5) augmented RC-diameter of the subtree itself.

The first three characteristics are inherited from the single-source problem [15] and are scalars. 4) considers the case where one or more source is *internal* to T_v, and one or more sink is *external* to T_v; 5) considers the case in which there are sources *and* sinks internal to T_v.

To motivate the issues involved in cases 4 and 5 above, consider the example in Fig. 3. In the figure, we have arrived at vertex v and need to be able to assess the maximum arrival time at v from u and w and the internal augmented path delays u to w and w to u. The critical observation is that these values depend on the capacitance of the tree *outside* of T_v (i.e., T/T_v). Of course, this *external capacitance* c_E is unknown at this stage of the algorithm. To assess the arrival time at v from terminal u, consider the case where $c_E = 0$; calculating the driver delay and interconnect delay of wire (u, v), we give the arrival time at v from u as

$$a(u, v) = a(u) + 2(2 + 4 + 1) + 5(1 + 4 + 1) = 204.$$

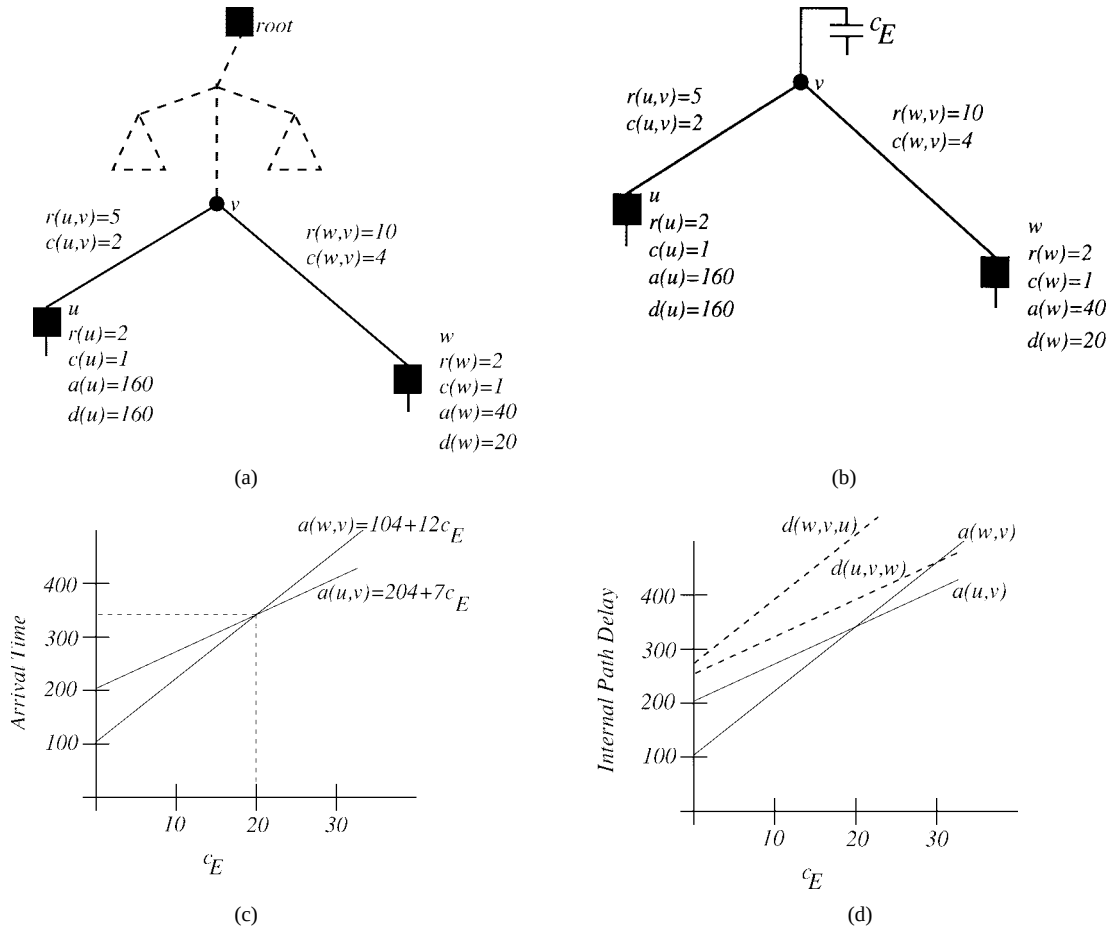


Fig. 3. (a) Example bottom-up computation, (b) abstraction of external capacitance c_E, (c) corresponding arrival time functions at v from u and w, and (d) augmented delay of internal paths.

Similarly, the arrival time at v via source w is given by $a(w, v) = 104$ (again, assuming $c_E = 0$). Thus, in general (i.e., for arbitrary c_E), we have

$$a(u, v) = 204 + 7c_E \quad \text{and} \quad a(w, v) = 104 + 12c_E$$

since the bottom-up accumulated resistance from u to v is seven and that from w to v is 12. Looking at Fig. 3(c), we see that not only does the arrival time depend on c_E, but so does the *critical source*—w dominating when $c_E < 20$ and u dominating when $c_E > 20$. Thus, the PWL *maximum* of $a(u, v)$ and $a(w, v)$ shown in Fig. 3(c) captures the arrival time at v from its descendants as a function of external capacitance c_E.

We must also consider the *internal* paths (u, v, w) and (w, v, u) as in Fig. 3(d). Consider the path (u, v, w); the augmented RC delay of this path includes $a(u, v)$, the scalar delay from v to sink w ($PD(v, w)$), and $d(w)$. Thus, noting that $PD(v, w) = 30$ and $d(w) = 20$, we have

$$d(u, v, w) = a(u, v) + PD(v, w) + d(w) = 254 + 7c_E.$$

Similarly, $d(w, v, u) = 274 + 12c_E$. These are shown by dashed lines in part (d) of the figure and amount to simply adding scalars to the y-intercepts of $a(u, v)$ and $a(w, v)$. Taking the piece-wise maximum of $d(u, v, w)$ and $d(w, v, u)$ gives us the *internal augmented RC diameter* of T_v [notice in this case however that $d(w, v, u)$ dominates $d(u, v, w)$ *for all* values of c_E].[10]

This discussion should give an idea of our approach: as we proceed bottom-up, we accumulate path resistance in the form of slopes in the PWL segments. As we proceed and consider various options such as repeater insertion, appropriate PWL operators are applied.

B. Solution Characterization

The formal characterization of a solution s for subtree T_v is given by the following parameters (using C-like structure syntax to refer to the parameters associated with a solution s).

- $s.\text{cost}$ scalar cost;
- $s.\text{cap}$ scalar capacitance;
- $s.d$ scalar augmented delay to sinks in T_v
- $s.f_A$ PWL giving arrival time at v from sources in T_v as a function of c_E;
- $s.f_I$ PWL giving augmented RC diameter for source/sink pairs internal to T_v.

[10] Note that this kind of decomposition is not possible in the case of the "arbitrary pair-wise constraints" version of the problem where the critical source of the subtree cannot be so easily determined—e.g., it is not the case that every *external* sink will have the same critical source as in our problem.

For each vertex v in the reoriented tree we compute a set $S(v)$ of *option* solutions for T_v where each $s \in S(v)$ is characterized as above.

C. PWL Primitives

In this section we define a variety of such PWL operators. We first give a formal definition of a PWL itself.

Definition 4.1: A PWL function f is a set of quadruples (y_0, slope, x_l, x_r) where each such quadruple represents a line segment and gives the y-intercept (y_0), *slope* and domain for which the segment is defined (i.e., all x where $x_l \leq x \leq x_r$). Additionally, no two segments may have overlapping domains.

As is typical in most applications of PWL functions, we store line segments in a linked list ordered by domain.

Equation (3), shown at the bottom of the page, contains a list of the PWL primitives we require. All of these operations can be performed in time linear in the number of segments in the participating functions by appropriately stepping through the participating segment lists.

D. Solution Dominance

A key issue in virtually all multidimensional dynamic programming applications is the elimination of provably suboptimal solutions and repeater insertion is no exception. The issue a rises for example in single-source buffer insertion [15] as well as other applications such as floor planning. In the typical situation, a solution s is characterized by a set of k *scalars* $s[1], s[2], \cdots, s[k]$; assume without loss of generality that minimization in each of these k dimensions is preferred. This induces a partial order $\preceq$ on a set of solutions S; for $s_1, s_2 \in S$, we have the following:

$$s_1 \preceq s_2 \iff s_1[i] \leq s_2[i] \; \forall \, i \in \{1 \cdots k\}.$$

In the context of multidimensional dynamic programming, we need not consider any solution $s_2 \in S$ if there exists $s_1 \in S$ such that $s_1 \preceq s_2$ (assume w.l.o.g. all solutions are distinct). Thus, for a given solution set S, we compute the *minima* of S.

Definition 4.2: The *minimal* or *dominant* subset of a set of k-dimensional points S is the largest subset $S' \subseteq S$ such that, for all $s_1, s_2 \in S'$, $s_1 \not\preceq s_2$, and $s_2 \not\preceq s_1$.

The problem of finding a minimal point set has a long history dating back to [14]. A variety of algorithms have been proposed; some achieve excellent asymptotic complexity (e.g., [14]) while others have been tuned for ease of implementation or fast *expected* run time (e.g., [3]). Implicitly, [15] solved a three-dimensional minima problem.

The reader may have noted by now that because of the PWL functions $s.f_A$ and $s.f_I$ a subsolution s in the repeater insertion problem does not correspond to a single point in k-dimensional space; rather, it corresponds to an infinite set of five-dimensional points—one for every value of c_E. One way to deal with this complication would be to *discretize* the c_E domain and thus introduce a different point-dominance problem for each discrete value of c_E (and introduce a degree of error depending on the granularity of discretization). We have not adopted the discretization approach; instead, we introduce the notion of a minimal functional subset (MFS).[11]

Definition 4.3: Let S be a set where $s \in S$ is a k-tuple of PWL functions of a variable x. For each $s \in S$, let s' be defined (for $i \in \{1 \cdots k\}$) as

$$s'[i](x) = \begin{cases} \infty, & \text{if } \exists s_0 \in S \text{ s.t. } s_0 \neq s, \text{ and } s_0[j](x) \leq s[j](x) \; \forall \, j \in \{1 \cdots k\} \\ s[i](x), & \text{o.w.} \end{cases}$$

The MFS of S is S':

$$S' = \{s' | \exists x \quad \text{s.t.} \quad s'[i](x) \neq \infty\}$$

Implicitly, if $s[i]$ is determined to be nonminimal for some value of x, all other $s[j]$'s must also be nonminimal for that same value of x. Note also that PWL functions in S' may no longer be "connected" since they may have suboptimal regions of the domain (i.e., ∞) separating useful regions of the domain. Finally, we point out that the fundamental pruning operation of the definition—for a given s and s_0, detect all ranges of x for which s_0 dominates s and alter the PWL's of s accordingly—can be implemented in time linear in k and the total number of line segments in question. This is done by first detecting the regions in which $s_0[i]$ dominates $s[i]$ (similar to a PWL_Max operation) for each i and then computing the intersection of these regions and finally updating the PWL's accordingly.

In our application, solutions are characterized by three scalars (degenerate PWL functions for the purpose of discussion) and two PWL functions. At each stage of the dynamic programming process, we would like to efficiently compute the MFS of a solution set S. We have devised an easy to implement algorithm for solving this problem which appears to works well in practice; it appears in pseudocode in Fig. 4. The algorithm takes a divide and conquer approach motivated by the following intuition. Suppose that the FMS of S is much smaller than S itself—i.e., many suboptimal solutions can be eliminated.[12] The hope is that many of the suboptimal solutions will be discarded at relatively deep levels of the recursion and thus we can avoid pair-wise comparisons at

[11] We use the notion of *subset* somewhat loosely here, however, the meaning should be clear.

[12] From experience, this is frequently the case; particularly when constructing solutions at a branch point from its children.

$$\begin{aligned} f \leftarrow \text{PWL\_Max}(f_1, f_2) &\iff f(x) = \max(f_1(x), f_2(x)) \quad \forall \, x \\ f \leftarrow \text{PWL\_AddScalar}(f_1, dy) &\iff f(x) = f_1(x) + dy \quad \forall \, x \\ f \leftarrow \text{PWL\_ShiftLeft}(f_1, dx) &\iff f(x) = f_1(x + dx) \quad \forall \, x \\ f \leftarrow \text{PWL\_AddSlope}(f_1, ds) &\iff f(x) = f_1(x) + x \cdot ds \quad \forall \, x \end{aligned} \tag{3}$$

```
Routine: MinimalFS_DC(S)
Input:   Solution set S
Output:  The minimal functional subset of S
  1. IF (|S| = 1)
  2.     RETURN S
  3. Divide S into equal sized sets S_l and S_r
  4. S'_l ← MinimalFS_DC(S_l)
  5. S'_r ← MinimalFS_DC(S_r)
  6. Compute S' = MFS(S'_l ∪ S'_r) by pair-wise comparison between S'_l and S'_r
  7. RETURN S'
```

Fig. 4. Computation of MFS(S) by divide and conquer.

```
Routine: MSRI(v)
Input:   Vertex v in oriented topology T
Purpose: Computes the minimal solution set S(v)
           among repeater assignments to T_v
  IF (v is a leaf)
    S(v) ← LeafSolutions(v)
  ELSE IF (v is a Steiner node)
    S(v) ← ∅
    FOREACH (child u of v)
      MSRI(u)
      S'(u) ← Augment(S(u), (v, u))
      IF (S(v) ≠ ∅)
        S(v) ← JoinSets(S(v), S'(u))     /* Combine solutions from subtrees */
        S(v) ← MinimalFS_DC(S(v))
      ELSE
        S(v) ← S'(u)
  ELSE IF (v is a candidate insertion point (degree-2))
    Let u be the child of v
    S'(u) ← Augment(S(u), (v, u))
    S(v) ← S'(u) ∪ RepeaterSolutions(S'(u))
    S(v) ← MinimalFS_DC(S(v))
  ELSE       /* v is the root */
    Let u be the child of v        /* Recall: terminals are degree-1 */
    S'(u) ← Augment(S(u), (v, u))
    S(v) ← RootSolutions(S'(u))
    S(v) ← MinimalFS_DC(S(v))
```

Fig. 5. Top-level algorithm for optimal multisource repeater insertion.

higher levels of the recursion. Thus, the approach targets fast performance in practice but retains an $\Omega(|S|^2)$ asymptotic worst case complexity in terms of the number of pair-wise comparisons.[13]

E. The Overall Algorithm

Finally, we present a high-level view of our algorithm for optimal repeater insertion in Fig. 5. We recursively traverse the tree and inductively compute solutions sets $S(v)$ from the solution sets of v's children. The top-level code invokes different subroutines depending on whether the current vertex v is a leaf, Steiner (branch) node, insertion point or the root itself; these routines appear in Figs. 6–9, respectively. Additionally, a routine for extending a solution set for a tree rooted at vertex v to a solution set that same tree augmented by the wire to v's parent appears in Fig. 10.

The algorithm follows the intuition presented in the preceding sections. Consider, for example, the subroutine *RepeaterSolutions* in Fig. 8. We visit each unbuffered solution s and candidate repeater b and determine the characteristics of a solution s' in which b is placed at the root of the unbuffered solution (assume as in the figure that the A-side of b connects to the parent and the B-side connects to the unbuffered solution s). First, we must add the cost of the additional repeater b; next, b decouples downstream capacitance and thus, b determines the new capacitance; next $s'.d$ must take the delay of b from A to B into account. Again, because of the decoupling effect of b, the arrival time at the B-side of b is precisely known—i.e., since we know that $c_E = c_{BA}(b)$, we simply evaluate $s.f_A(c_{BA}(b))$; $s'.f_A$ must now reflect the situation on the A-side of b which is accomplished by adding the intrinsic delay and creating a new PWL with slope equal to $r_{BA}(b)$. Finally, we notice that $s'.f_I$ is now completely determined by $c_{BA}(b)$ and independent of c_E seen at the A-side of b. The other subroutines follow this same kind of reasoning.

[13] Note however, that we have not bounded the size of the PWL functions themselves (i.e., number of segments). In the worst case, the PWL functions can grow exponentially in the number of insertion points. However, such degenerate scenarios appear to occur infrequently in practice.

```
Routine: LeafSolutions(v)
Input:   Leaf (terminal) vertex v
Output:  The the (singleton) solution set for v
  s.cost ← 0
  s.cap ← c(v)
  s.d ← d(v)
  s.f_A ← a(v) + r(v) c_E     /* Single segment PWL with y-int at a(v) and slope r(v) */
  s.f_I ← −∞                  /* No internal paths */
  RETURN {s}
```

Fig. 6. *LeafSolutions()* subroutine for optimal repeater insertion.

```
Routine: JoinSets(S_1, S_2)
Input:   Solution sets S_1 and S_2 from disjoint subtrees
Output:  The minimal solution set S when the
           subtrees are joined at a common parent
  S ← ∅
  FOREACH (pair of solutions  s_1 ∈ S_1, s_2 ∈ S_2)
    s.cost ← s_1.cost + s_2.cost                  /* Scalar operations first */
    s.cap ← s_1.cap + s_2.cap
    s.d ← max(s_1.d, s_2.d)
    f^1_A ← PWL_ShiftLeft(s_1.f_A, s_2.cap)       /* Signals from left now see cap on right */
    f^2_A ← PWL_ShiftLeft(s_2.f_A, s_1.cap)       /* Signals from right now see cap on left */
    s.f_A ← PWL_Max(f^1_A, f^2_A)                 /* Combined arrival time functions */
    f^12_I ← PWL_AddScalar(f^1_A, s_2.d)          /* New left-to-right internal paths */
    f^21_I ← PWL_AddScalar(f^2_A, s_1.d)          /* New right-to-left internal paths */
    f^v_I ← PWL_Max(f^12_I, f^21_I)               /* v is the current root */
    f^1_I ← PWL_ShiftLeft(s_1.f_I, s_2.cap)       /* Account for old internal paths on left */
    f^2_I ← PWL_ShiftLeft(s_2.f_I, s_1.cap)       /* Account for old internal paths on right */
    s.f_I ← PWL_Max(f^v_I, PWL_Max(f^1_I, f^2_I)) /* Put it all together */
    S ← S ∪ {s}
  RETURN S
```

Fig. 7. The *JoinSets()* subroutine for optimal repeater insertion.

```
Routine: RepeaterSolutions(S)
Input:   Unbuffered solution set S
Output:  A solution set in which all solutions are buffered with
           an oriented repeater from the given library L_b
  S' ← ∅
  FOREACH (s ∈ S and b ∈ L_b)
    /* Assume that parent connects to the A-side of b; child to B-side */
    s'.cost ← s.cost + s(b)
    s'.cap ← c_AB(b)
    s'.d ← s.d + i_AB(b) + r_AB s.cap
    a ← s.f_A(c_BA(b)) + i_BA(b)
    s'.f_A ← a + r_BA(b) · c_E          /* accumulated resistance below b not affected by c_E */
    s'.f_I ← s.f_I(c_BA(b))             /* internal paths independent of c_E */
    Compute s'' in the same way for opposite orientation of b
    S' ← S' ∪ {s', s''}
  RETURN S'
```

Fig. 8. The *RepeaterSolutions()* subroutine for optimal repeater insertion.

Routine: RootSolutions(S)
Input: Solution set S which considers the entire tree *except* the root driver itself
Output: A solution set taking the root driver into account. Since these are global solutions, only the minimal solutions with respect to cost and final diameter $ARD(T)$

$S' \leftarrow \emptyset$
FOREACH ($s \in S$)
  $s'.cost \leftarrow s.cost$
  $d \leftarrow s.d + r(v)\ s.cap + a(v)$ /* root as source case */
  $d' \leftarrow s.f_A(c(v)) + d(v)$ /* root as sink case */
  $d_I \leftarrow s.f_I(c(v))$ /* root not involved */
  $s'.d \leftarrow \max(d, d', d_I)$ /* store $ARD(T)$ in $s.d$ */
  /** Note: $s'.cap$, $s'.f_A$ and $s'.f_I$ do not matter **/
  $s'.cap \leftarrow s'.f_A \leftarrow s'.f_I \leftarrow -\infty$
  $S' \leftarrow S' \cup \{s\}$
Delete from S' all non-minimal solutions w.r.t $s.cost$ and $s.d$
  /* Easy in two dimensions */
RETURN S'

Fig. 9. The *RootSolutions()* subroutine for optimal repeater insertion.

Routine: Augment($S, (u, v)$)
Input: Solution set S and the wire (u, v) by which we are extending the subtree
Output: A solution set built from S set taking wire (u, v) into account

$S' \leftarrow \emptyset$
FOREACH ($s \in S$)
  $s'.cost \leftarrow s.cost$
  $s'.cap \leftarrow s.cap + c(u, v)$
  $s'.d \leftarrow s.d + r(u, v)(\frac{c(u,v)}{2} + s.cap)$
  $f'_A \leftarrow$ PWL_AddScalar($s.f_A, r(u, v)(\frac{c(u,v))}{2})$)
  $s'.f_A \leftarrow$ PWL_AddSlope($f'_A, r(u, v)$)
  $s'.f_I \leftarrow$ PWL_ShiftLeft($s.f_I, c(u, v)$)
  $S' \leftarrow S' \cup \{s\}$
RETURN S'

Fig. 10. The *Augment()* subrountine for optimal repeater insertion.

Finally, optimality of the algorithm follows inductively from the bottom-up structure of the algorithm.

Theorem 4.1: Consider $S(v)$ as computed by the multisource repeater insertion (MSRI) algorithm in Fig. 5. Let $s(c_E)$ be the five-tuple of scalars (s.cost, s.cap, $s.d$, $s.f_A(c_E)$, $s.f_I(c_E)$). The following property holds:

$$s \in S(v) \Longleftrightarrow \exists c_E \text{ s.t. } \forall \text{ repeater assigments } s' \neq s \text{ to } T_v, \exists c_E \text{ s.t. } s'(c_E) \not\prec s(c_E).$$

V. Discussion

A number of issues with respect to the repeater insertion algorithm are worth mentioning.

- By considering a discretization of the various dimensions of the repeater insertion problem, it is possible to give a *pseudopolynomial* bound on the algorithms run-time. The practical utility of such a bound is limited and therefore, we defer to the experimental results of the next section for more meaningful evaluation of run-time.
- The algorithm can also solve the driver sizing problem subject to the assumption that drivers are single input (thus allowing us to easily take into account the affect a source driver has on its preceding stage).
- Organizational conventions such as maintaining solution sets in sorted order by cost and secondarily by capacitance can potentially improve performance by allowing us to easily avoid unnecessary comparisons when finding a minimal solution set.
- Many different strategies for finding minimal solution sets can be imagined and are perhaps worth exploring.
- An extension allowing the use of inverters as repeaters is possible and straightforward.

TABLE I
Technology Parameters Used in Experiments. Bidirectional Repeaters and Source/Sink Drivers Are Constructed from a Pair of Unidirectional Buffers

Interconnect Parameters	
$\Omega/\mu m$	0.12
$fF/\mu m$	0.15

Unidirectional Buffer/Driver Parameters	
"Cost"	1.0
Intrinsic Delay	$100.0ps$
Input Cap	$0.05pF$
Output Resistance	$800(\Omega)$

VI. Experiments

We have implemented prototypes of the algorithms described in this paper and we report preliminary results in this section. Since the algorithm assures optimality under delay models accepted and successful in single-source optimization, perhaps the most important experimental result is the demonstration of the tractability of the algorithm. Nevertheless, we do report timing results based on the technology parameters used.

Table I presents the technology parameters for the experiments; the parameters are the same as those used in [20] and are, we believe, representative of typical submicron technologies. Experimental results appear in Table II. In this set of experiments, we assume that all terminals serve as both sources and sinks and that all arrival times and downstream delay times are zero—i.e., *unaugmented* RC-diameter is the performance measure. We generated a set of ten random point sets with ten terminals on a 1 cm $\times$ 1cm grid; we did the same for 20 terminals. Steiner trees connecting the terminals were then generated using the P-Tree$_A$ algorithm [16]. Repeater insertion points were then added to the topology such that consecutive insertion points were no more than approximately 800 μm apart.[14]

The repeater insertion algorithm was then applied to the topology using the repeater formed by a pair of the buffers described in Table I. We then ran the algorithm in a *driver sizing* mode on the same topology using a driver library derived from the basic buffer component in Table I: this was done by considering the buffer in the table as a "1X" driver and then introducing 2X, 3X, and 4X buffers [where a kX buffer has cost k, resistance $(800/k)$ Ω, and capacitance k(0.05) pF]. These three buffers were used to effectively make a library 9 terminal drivers (when orientation is considered). Additionally, it was assumed that the previous stage resistance of each terminal was 400 Ω and that the subsequent stage capacitance

[14]We also ensured that all wire segments contained at least one insertion point. As a result, the average distance between insertion points was considerably less than 800 μm—approximately 450 μm.

TABLE II
EXPERIMENTAL RESULTS. COLUMNS 1 AND 2 REPORT THE NET SIZE AND THE AVERAGE NUMBER OF REPEATER INSERTION POINTS. RESULTS IN COLUMNS 3–7 ARE AVERAGES OF VALUES NORMALIZED TO THE CORRESPONDING VALUES FOR MIN-COST SOLUTIONS (I.E., NO REPEATER INSERTION OR SIZING). COLUMNS 3 AND 4 REPORT THE AVERAGE MINIMAL DIAMETER AND ASSOCIATED COST ACHIEVED BY DRIVER SIZING; ACHIEVABLE RC-DIAMETER VIA DRIVER SIZING ALONE AND THE ASSOCIATED COST. COLUMN 5 GIVES THE LOWEST COST REPEATER INSERTION SOLUTION WHICH EQUALLED OR BETTERED THE DIAMETER OF THE CORRESPONDING DRIVER SIZED SOLUTION. FINALLY, COLUMNS 6 AND 7 GIVE THE RESULTS FOR THE MIN DIAMETER REPEATER INSERTION PROBLEM. TEN RANDOM NETS ON A 1-cm × 1-cm GRID WERE USED FOR EACH CARDINALITY

	Avg. #	D.S.: Min ARD		R.I.: Min Cost s.t.	Min ARD R.I.	
$\|N\|$	i-points	D_{ds}	Cost	$ARD \leq D_{ds}$	D	Cost
10	30.4	0.73	1.45	1.18	0.55	1.73
20	36.6	0.77	1.26	1.06	0.44	1.51

TABLE III
COMPARISON OF FASTEST DRIVER SIZING AND REPEATER INSERTION SOLUTIONS FOR SIX SAMPLE TOPOLOGIES. COST IS GIVEN IN NUMBER OF EQUIVALENT 1X BUFFERS

Topo	N	Wire-Len (μm)	DS Diameter(ps)	DS Cost	RI Diameter(ps)	RI Cost
T1	10	26.2K	5786	32	3959	34
T2	10	24.0K	6069	31	4810	32
T3	10	26.2K	6263	34	4740	34
T4	20	31.9K	8439	53	5088	54
T5	20	26.7K	13296	45	6551	62
T6	20	31.3K	12125	62	7531	62

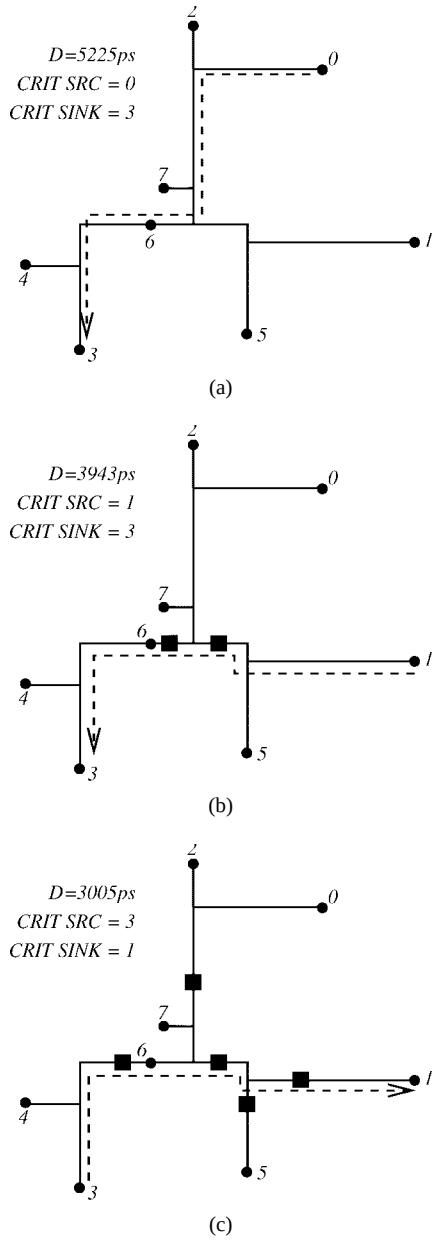
of each terminal was 0.2 pF (the same assumption was made for repeater insertion).

As may be expected, far more substantial reductions in RC-diameter were observed for repeater insertion. Additionally, when the minimum RC-diameter achievable by driver sizing is used as a constraint for repeater insertion, we observe substantially lower cost overhead for equivalent or better diameter. Note that columns 3–7 are normalized to the corresponding values for the min-cost solution. For example, on ten-pin nets, driver sizing reduced RC-diameter to 73% that of the min-cost solution on average while repeater insertion achieved an average of 55%. We also illustrate some particular solutions in Table III. These give the highest performance solutions produced by both sizing and repeater insertion.

We illustrate solutions produced by the algorithm in Fig. 11. The example is an eight-pin net where all pins can act as either drivers or sinks and the performance measure is again RC-diameter. The reader can observe how performance is improved with added buffering resources and also how the critical input-to-output path changes as the algorithm carefully balances the requirements of all paths.

Table IV gives average CPU times. As stated in the previous section, empirical evidence is the best way to judge the tractability of algorithms such as those proposed here. Indeed, the achieved run-times are quite reasonable and were derived from largely unoptimized prototype code.[15]

[15]Experiments have also been run on examples with closer spacing of insertion points and therefore higher complexity. These results were not included because the improvement in solution quality versus wider spacing of insertion points was small; however these results were typically obtained within a few minutes on a Sun SPARC 10 workstation (e.g., 20 pins, 300 μm average insertion point spacing).



(a) (b) (c)

Fig. 11. (a) Optimization of an eight-pin net. Total wire length is 19.6 K μm. (a) Unoptimized topology, (b) two-repeater solution constructed by repeater insertion algorithm, and (c) five-repeater solution constructed by repeater insertion algorithm. In each case, the resulting RC-diameter is given as well as the critical source and sink in each case. Delays are computed using Elmore.

TABLE IV
AVERAGE RUN-TIMES IN CPU SECONDS ON A SUN SPARC 10 WORKSTATION

$\|N\|$	Driver Sizing	Repeater Insertion
10	0.9	6.9
20	6.0	28.9

VII. CONCLUSIONS/COMMENTS

It is clear that interconnect optimization will play a central role in future CAD tools for physical design; optimization of multisource nets is no exception to this trend. As such, we propose that the techniques presented in this paper are of fundamental interest. First, the notion of the augmented RC-diameter gives a natural and practical performance metric for

multisource net optimization. Second, we have characterized multisource optimization via manipulation of PWL functions and, from this, proposed algorithm demonstrating that optimally applying the powerful technique of repeater insertion is indeed a tractable problem.

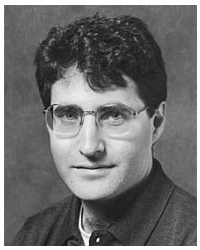
Future directions this research may take include exploration of more sophisticated algorithmic speedup techniques (e.g., advanced pruning techniques). From a nonalgorithmic (or designer) perspective, topics such as further evaluation of the tradeoffs between driver sizing and repeater insertion and the effects of asymmetric source/sink distributions are also of interest. Further, we note that there is no fundamental reason why the basic techniques introduced here cannot be utilized to solve other optimization problems in multisource nets such as wire sizing and topology synthesis (e.g., given the results in this paper, a multisource version of the P-Tree timing-driven Steiner router [16] is now possible). We are currently pursuing projects to evaluate the tractability of such methods.

ACKNOWLEDGMENT

The authors would like to thank Prof. M. Sarrafzadeh of Northwestern University, Evanston, IL, for pointing them toward the literature on the point dominance problem. They also wish to thank Dr. K. Clarkson of AT&T Bell Laboratories, Murray Hill, NJ, for pointing out [3] and additional input on point dominance problems.

REFERENCES

[1] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990, pp. 212–215.
[2] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Fidelity and near-optimality of Elmore-based routing constructions," presented at *IEEE Int. Conf. Computer-Aided Design,* 1993.
[3] K. L. Clarkson, "More output-sensitive geometric algorithms," in *Proc. 35th Annu. Symp. on Foundations of Computer Science,* 1994, pp. 695–702.
[4] J. J. Cong and K. S. Leung, "Optimal wiresizing under Elmore delay model," *IEEE Trans. Computer-Aided Design,* vol. 14 no. 3, pp. 321–336, 1995.
[5] J. J. Cong, K. S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," in *Proc. ACM/IEEE Design Automation Conf.,* 1993, pp. 606–611.
[6] J. J. Cong and P. H. Madden, "Performance driven routing with multiple sources," in *Proc. IEEE Symp. on Circuits and Systems,* 1995, pp. 203–206.
[7] J. J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1995, pp. 568–574.
[8] ———, "An efficient approach to simultaneous transistor and interconnect sizing," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1996, pp. 181–186.
[9] S. Dhar, "Delay optimization algorithms for tree models of MOS circuits," Ph.D. dissertation, Dept. of Electrical Engineering, Washington University, Stevens Institute of Technology, Sept. 3, 1987.
[10] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers," *J. Appl. Phys.,* vol. 19, pp. 55–63, 1948.
[11] X. Hong, T. Xue, E. S. Kuh, C. K. Cheng, and J. Huang, "Performance-driven Steiner tree algorithms for global routing," in *Proc. ACM/IEEE Design Automation Conf.,* June 1993, pp. 177–181.
[12] T. B. Huang and Y. C. Jeng *et al.*, "Bidirectional bus repeater," U.S. Patent 5 202 593, Apr. 13, 1993.
[13] D. Y. Kao, C. C. Tsai, C. K. Cheng, and T. T. Lin, "New design and implementation for signal repeaters," in *VLSI/CAD Workshop,* Taiwan, Aug. 17–19, 1995, pp. 173–176.
[14] H. T. Kung, F. Luccio, and F. P. Preperata, "On finding the maxima of a set of vectors," *J. ACM 22,* Oct. 4, 1975, pp. 469–476.
[15] J. Lillis, C.-K. Cheng, and T.-T. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *IEEE J. Solid-State Circuits,* vol. 31, no. 3, pp. 437–447, Mar. 1996.
[16] J. Lillis, C.-K. Cheng, T.-T. Lin, and C.-Y. Ho, "New techniques for performance driven routing with explicit area/delay tradeoff and simultaneous wire sizing," in *Proc. 33rd ACM/IEEE Design Automation Conf.,* Las Vegas, June 1996, pp. 395–400.
[17] J. Lillis, C.-K. Cheng, and T.-T. Lin, "Simultaneous routing and buffer insertion for high performance interconnect," in *Proc. 6th IEEE Great Lakes Symp. on VLSI,* Ames, IA, Mar. 1996, pp. 148–153.
[18] T. M. Lin and C. A. Mead, "Signal delay in general RC-networks," *IEEE Trans. Computer-Aided Design,* vol. CAD-3, pp. 331–349, 1984.
[19] N. Menezes, Intel Corp., personal communication, Oct. 1996.
[20] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1996, pp. 44–49.
[21] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design,* vol. 2, no. 3, pp. 202–211, 1983.
[22] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.,* 1994, pp. 387–391.
[23] ———, Iowa State University, personal communication, Oct. 1996.
[24] C. C. Tsai, D. Y. Kao, and C. K. Cheng, "Performance driven bus buffer insertion," *IEEE Trans. Computer-Aided Design,* pp. 429–437, Apr. 1996.
[25] R. S. Tsay, "Exact zero skew," in *Proc. IEEE Int. Conf. Computer-Aided Design,* 1991, pp. 336–339.
[26] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. Int Symp on Circuits and Systems,* 1990, pp. 865–868.
[27] A. Vittal and M. Marek-Sadowska, "Minimal delay interconnect design using alphabetic trees," in *Proc. ACM/IEEE Design Automation Conf.,* 1994, pp. 392–396.



John Lillis received the B.S. degree in computer science from the University of Washington, Seattle, in 1989 and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego in 1992 and 1996, respectively.

From 1996 to 1997, he was a National Science Foundation (NSF) Postdoctoral Visitor at the University of California at Berkeley. In 1997, he joined the University of Illinois at Chicago, where he is currently an Assistant in the Department of Electrical Engineering and Computer Science. His research interests are in the areas of design automation for integrated circuits and combinatorial optimization.

Chung-Kuan Cheng (S'82–M'84–SM'95) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taiwan, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, in 1984.

From 1984 to 1986, he was a Senior CAD Engineer at Advanced Micro Devices Inc., Sunnyvale, CA. In 1986, he joined the University of California, San Diego, where he is currently a Professor in the Computer Science and Engineering Department, and an Adjunct Professor in the Electrical and Computer Engineering Department. His research interests include network optimization and design automation on microelectronics circuits.

He is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS since 1994. He is a recipient of the Best Paper Award, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN 1997, the NCR Excellence in Teaching Award, School of Engineering, UCSD, 1991.