An Efficient and Optimal Algorithm for Simultaneous Buffer and Wire Sizing

Chris C. N. Chu and D. F. Wong, *Member, IEEE*

***Abstract*—In this paper, we consider the problem of interconnect delay minimization by simultaneous buffer and wire sizing under the Elmore delay model. We first present a polynomial time algorithm SBWS to minimize the delay of an interconnect wire. Previously, no polynomial time algorithm for the problem has been reported in the literature. SBWS is an iterative algorithm with guaranteed convergence to the optimal solution. It runs in quadratic time and uses constant memory for computation. Experimental results show that SBWS is extremely efficient in practice. For example, for an interconnect of 10 000 segments and buffers, the CPU time is only 0.255 s. We then extend our result to handle interconnect trees. We present an algorithm SBWS-T which always gives the optimal solution. Experimental results show that SBWS-T is faster than the greedy wire sizing algorithm [2] in practice.**

***Index Terms*— Buffer sizing, interconnect, performance optimization, physical design, wire sizing.**

I. Introduction

In the past, gate delay was the dominating factor in circuit design. However, as the feature size of VLSI devices continues to decrease, interconnect delay becomes increasingly important. Nowadays, feature size has been reduced to 0.25 μm in advance technologies. Interconnect delay has become the dominating factor in determining system performance. In many systems designed today, as much as 50%–70% of clock cycle is consumed by interconnect delay [10]. It is predicted in the National Technology Roadmap for Semiconductors [19] that the feature size will be reduced to 0.13 μm by 2003 and 0.07 μm by 2009. So we expect the significance of interconnect delay will further increase in the near future.

Wire sizing was first shown by Cong and Leung [12] to be an effective technique to reduce interconnect delay. They proposed the greedy wire sizing algorithm (GWSA) which minimizes the weighted sink delay of an interconnect tree. Since then, many wire sizing results were published. Some examples are a closed form formula for an interconnect wire [3], [15], its extension to an interconnect tree [4], an algorithm to minimize the maximum sink delay [18], and an algorithm for interconnects with multiple sources [8].

In [12], discrete wire sizing (i.e., the segment widths must be chosen from a given set of discrete choices) was considered. GWSA was later extended by Chen and Wong [2] to continuous wire sizing (i.e., the segment widths can be from a continuous range of real numbers). Recently, Chu and Wong [6] proved that GWSA for continuous wire sizing runs in time linear to the number of segments.

In order to reduce delay and to maintain signal integrity, usually buffers are inserted to interconnect wires. Sizing the buffers appropriately can also reduce the interconnect delay significantly. Since buffer sizes affect wire sizing solutions and wire sizes affect buffer sizing solutions, it is beneficial to simultaneously size both buffers and wires. The algorithm GWSA has been extended to handle simultaneous buffer and wire sizing in [11] for discrete sizing and in [1] for continuous sizing. These algorithms have been shown to be very efficient in practice. However, no bounds on the runtime of them are known. In Section V of this paper, we do some experiments with GWSA for continuous buffer and wire sizing. We observe that even for a single wire, the runtime is no longer linear (as in the case of wire sizing alone).

Some other related results on wire sizing and buffer sizing are listed below. Menezes *et al*. [17] applied the sequential quadratic programming approach to simultaneous gate and wire sizing. That is the sizing problem is reduced to a sequence of quadratic programming subproblems. No bound on the runtime of the algorithm was reported. Lillis *et al*. [16] gave an algorithm for simultaneous buffer insertion, buffer sizing and wire sizing based on dynamic programming. This algorithm runs in pseudopolynomial time and requires a substantial amount of memory. Chu and Wong [5] also considered simultaneous buffer insertion, buffer sizing and wire sizing. A closed form optimal solution was presented. However, in that paper, only wire area capacitance was considered. Wire fringing capacitance [20], which will become more and more significant as feature size decreases, was ignored. Taking wire fringing capacitance into account significantly complicates the problem and [5] can only give an approximate solution. Chu and Wong [7] showed that the simultaneous buffer insertion and wire sizing problem can be formulated as a convex quadratic program. The convex quadratic program has a small size and some special structures, and so can be solved very efficiently. The result was extended to handle buffer sizing by enumerating all possible combinations of buffer sizes. A pruning technique was proposed to improve the efficiency. A comprehensive survey on previous works can be found in [10].

In this paper, we consider the continuous version of the interconnect delay minimization problem by simultaneously sizing buffers and wire segments. We first consider the prob-

Manuscript received April 24, 1998; revised March 15, 1999. This paper was recommended by Associate Editor M. Pedram.

C. C. N. Chu is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011 USA (e-mail: cnchu@iastate.edu).

D. F. Wong is with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712 USA.

Publisher Item Identifier S 0278-0070(99)06623-3.

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

lem for a single interconnect wire. Basically, an interconnect wire joining a source and a sink is divided into some uniform-width wire segments. Some of the adjacent segments have buffers in between. The problem is to determine the buffer sizes and segment widths so that the Elmore delay from the source to the sink is minimized. The details of the problem formulation are presented in Section II. Note that no previous result can solve this problem optimally in a provably polynomial time bound. In particular, both wire area capacitance and wire fringing capacitance are taken into account in this paper. An approach completely different from that in [5] is required here.

We make the following contributions in this paper.

- We present an iterative algorithm SBWS, for the simultaneous buffer and wire sizing problem. We prove that SBWS always converges to the optimal solution.
- We prove that for an interconnect wire consisting of n buffers and segments, SBWS runs in $O(n^2 + n\log(1/\epsilon))$ time, where ϵ specifies the precision of computation (see Theorem 1). Since $\log(1/\epsilon)$ is bounded by the number of bits in the input, the total runtime is quadratic to the input size. This is the first polynomial time algorithm for the simultaneous buffer and wire sizing problem considered in this paper.
- SBWS requires only constant memory for computation.
- We show that our result can be extended to handle interconnect trees. We present an algorithm SBWS-T which always gives the optimal solution for the weighted sink delay objective.
- We demonstrate experimentally that SBWS and SBWS-T are both extremely efficient in practice. For SBWS, for an interconnect of 10 000 segments and buffers, the CPU time of SBWS is only 0.255 s. Besides, we observe that SBWS runs in linear time in practice. For SBWS-T, we show that it is faster than GWSA in practice.

The rest of the paper is organized as follows. In Section II, we present the formulation of the simultaneous buffer and wire sizing problem for a wire. In Section III, the algorithm SBWS, its optimality proof and its runtime analysis are presented. In Section IV, we describe how to extend our result to handle interconnect trees. In Section V, some experimental results to show the efficiency of SBWS and SBWS-T are presented.

II. Problem Formulation for a Wire

In this paper, a *component* means either a buffer or a wire segment. Given a source with driver resistance R_D, a sink with load capacitance C_L, the source and the sink are linked by an interconnect consisting of n components. The ith component is either a buffer of size x_i or a wire segment of width x_i. The simultaneous buffer and wire sizing problem is to minimize the delay from the source to the sink with respect to $x_1, \ldots, x_n$.

In order to simplify the notations, we treat the source and the sink as buffers of fixed size in this paper. Let the source be called the 0th component and the sink be called the $(n+1)$th component. Let m be the number of sizable buffers in the interconnect (i.e., excluding the source and the sink). For $0 \le j \le m+1$, let b_j be the component index

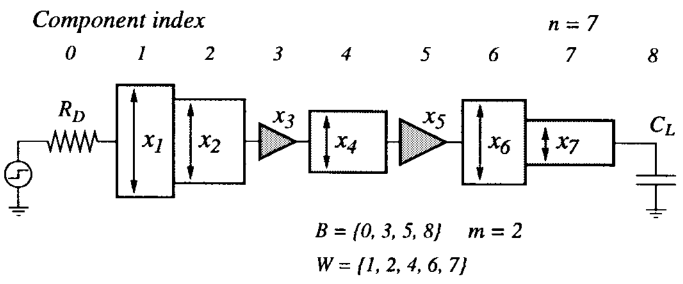

Fig. 1. The simultaneous buffer and wire sizing problem.

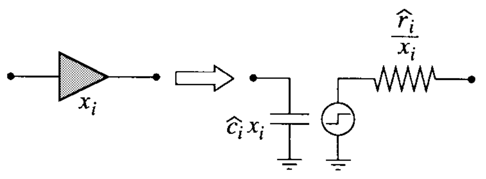

Fig. 2. The model of a buffer as a switch-level RC circuit.

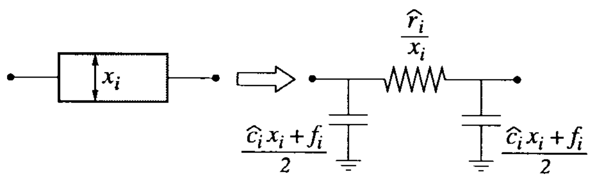
$\frac{\hat{r}_i}{x_i}$
x_i
$\frac{\hat{c}_i x_i + f_i}{2}$
$\frac{\hat{c}_i x_i + f_i}{2}$

Fig. 3. The model of a wire segment as a π-type RC circuit.

of the jth buffer. Note that $b_0 = 0$ and $b_{m+1} = n+1$. Let $\mathcal{B}$ be the set of component indexes of buffers (i.e., $\mathcal{B} = \{b_0, b_1, \ldots, b_m, b_{m+1}\}$). Let $\mathcal{W}$ be the set of component indexes of wire segments (i.e., $\mathcal{W} = \{0, 1, \ldots, n+1\} - \mathcal{B}$). See Fig. 1 for an illustration.

If component i is a buffer (i.e., $i \in \mathcal{B}$), then it is modeled as a switch-level RC circuit as shown in Fig. 2. The output resistance and the input capacitance of the buffer are $\hat{r}_i/x_i$ and $\hat{c}_i x_i$, respectively, where $\hat{r}_i$ and $\hat{c}_i$ are unit effective resistance and unit gate capacitance of the buffer, respectively. As we mentioned above, we treat the source (component 0) and the sink (component $n+1$) as fixed size buffers. So x_0, $\hat{r}_0$, x_{n+1}, and $\hat{c}_{n+1}$ are set to some arbitrary values such that $R_D = \hat{r}_0/x_0$ and $C_L = \hat{c}_{n+1}x_{n+1}$.

If component i is a wire segment (i.e., $i \in \mathcal{W}$), then it is modeled as a π-type RC circuit as shown in Fig. 3. The resistance and the capacitance of the wire segment are $\hat{r}_i/x_i$ and $\hat{c}_i x_i + f_i$, respectively, where $\hat{r}_i$, $\hat{c}_i$, and f_i are the unit width wire resistance, unit width wire area capacitance, and wire fringing capacitance of the segment, respectively.

For $0 \le i \le n$, if $b_j \le i < b_{j+1}$, let

$$R_i = \frac{\hat{r}_{b_j}}{x_{b_j}} + \sum_{k=b_j+1}^{i} \frac{\hat{r}_k}{x_k} \tag{1}$$

$$C_i = \hat{c}_{b_{j+1}} x_{b_{j+1}} + \sum_{k=i+1}^{b_{j+1}-1} (\hat{c}_k x_k + f_k). \tag{2}$$

Intuitively, R_i is the sum of all resistances before component $i{+}1$ (up to the last buffer), and C_i is the sum of all capacitances

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

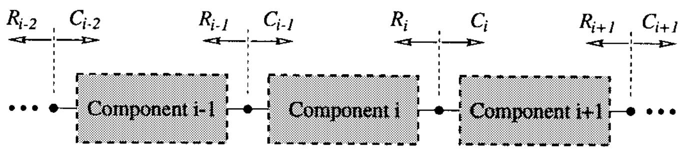

Fig. 4. Illustration of R_i and C_i.

after component i (up to the next buffer). See Fig. 4 for an illustration. Let the *upstream resistance* of component i be R_{i-1}. Let the *downstream capacitance* of component i be C_i if $i \in \mathcal{B}$, or $C_i + (\hat{c}_i x_i + f_i)/2$ if $i \in \mathcal{W}$.

In this paper, the widely used Elmore delay model [14] is used for delay calculation. Basically, the Elmore delay from the source to the sink is the sum of the delays associated with the components, where the delay associated with a component is equal to its resistance times its downstream capacitance. In other words, the Elmore delay from the source to the sink is given by

$$D = \sum_{j=0}^{m} \left(\frac{\hat{r}_{b_j}}{x_{b_j}} C_{b_j} + \sum_{i=b_j+1}^{b_{j+1}-1} \frac{\hat{r}_i}{x_i} \left(C_i + \frac{\hat{c}_i x_i + f_i}{2} \right) \right). \quad (3)$$

The problem is to minimize D with respect to $x_1, \ldots, x_n$.

III. The Algorithm SBWS

In this section, we derive a polynomial time algorithm SBWS for the simultaneous buffer and wire sizing problem presented in Section II. We first derive the necessary and sufficient conditions for optimality and write down the system of equations specifying the optimal solution. Then, we show how to solve the system of equations in polynomial time using a simple binary search technique.

The necessary conditions for optimality are $\partial D/\partial x_i = 0$ for $1 \le i \le n$. If $i \in \mathcal{B}$, then we can write D in (3) in terms of x_i as

$$D = R_{i-1}\hat{c}_i x_i + \frac{\hat{r}_i}{x_i} C_i + \text{terms independent of } x_i.$$

So $\partial D/\partial x_i = 0$ is equivalent to

$$\hat{c}_i R_{i-1} x_i^2 = \hat{r}_i C_i. \quad (4)$$

If $i \in \mathcal{W}$, then we can write D in (3) in terms of x_i as

$$D = R_{i-1}\hat{c}_i x_i + \frac{\hat{r}_i}{x_i} \left(C_i + \frac{f_i}{2} \right) + \text{terms independent of } x_i.$$

So $\partial D/\partial x_i = 0$ is equivalent to

$$\hat{c}_i R_{i-1} x_i^2 = \hat{r}_i (C_i + f_i/2). \quad (5)$$

Note that D is a posynomial [13] in $x_1, \ldots, x_n$. It is well known that under a variable transformation, a posynomial is equivalent to a convex function [13]. So D has a unique global minimum and no other local minimum. That means, if for some solution, $\partial D/\partial x_i = 0$ for $1 \le i \le n$, then the solution is optimal. In other words, (4) and (5) are both necessary and sufficient conditions for optimality.

Also, observe that for $0 \le i \le n$, (1) and (2) can be rewritten recursively as follows:

$$R_i = \hat{r}_i / x_i, \quad \text{if } i \in \mathcal{B} \quad (6)$$

$$R_i = R_{i-1} + \hat{r}_i / x_i, \quad \text{if } i \in \mathcal{W} \quad (7)$$

$$C_i = \hat{c}_{i+1} x_{i+1}, \quad \text{if } i+1 \in \mathcal{B} \quad (8)$$

$$C_i = C_{i+1} + \hat{c}_{i+1} x_{i+1} + f_{i+1}, \quad \text{if } i+1 \in \mathcal{W}. \quad (9)$$

As a result, finding the optimal solution to the problem is equivalent to solving (4) and (5) for $x_1, \ldots, x_n$, where $R_0, \ldots, R_n, C_0, \ldots, C_n$ satisfy (6)–(9). In other words, we need to solve a system of $3n+2$ nonlinear equations (4)–(9) for the variables $x_1, \ldots, x_n, R_0, \ldots, R_n$, and $C_0, \ldots, C_n$. We explore some special properties of the system and show how to solve the system in quadratic time below.

The basic idea is instead of considering the system of equations (4)–(9) directly, we consider a modified system obtained by adding an extra equation to fix the value of R_n and ignoring the equation $R_0 = \hat{r}_0/x_0$ [one of the equations in (6) when $i = 0$]. We show that this modified system of equations can be solved in linear time. Moreover, if the resulting R_0 equals $\hat{r}_0/x_0$ ($= R_D$ by definition), then the solution of the modified system will also be a solution of the original system, and hence the optimal solution of the simultaneous buffer and wire sizing problem. In the following, we first show how to solve the modified system in linear time. Then we show how to find the value of R_n such that the resulting R_0 equals R_D.

For any wire segment i, the lemma below gives the value of x_i if R_i and C_i are known.

Lemma 1: For any $i \in \mathcal{W}$, for the solution of the modified system

$$x_i = \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_i R_i (C_i + f_i/2)}}{2\hat{c}_i R_i}.$$

Proof: Eliminating R_{i-1} from (5) and (7), we have $\hat{c}_i(R_i - \hat{r}_i/x_i)x_i^2 = \hat{r}_i(C_i + f_i/2)$, or equivalently, $\hat{c}_i R_i x_i^2 - \hat{r}_i\hat{c}_i x_i - \hat{r}_i(C_i + f_i/2) = 0$. Solving the quadratic equation and taking the positive root, we get the result. ■

So if we know R_i and C_i for some i between 1 and n, then by Lemma 1 (if $i \in \mathcal{W}$) or by (6) (if $i \in \mathcal{B}$), we can determine x_i, and hence R_{i-1} and C_{i-1}, in constant time. Since R_n is fixed by the extra equation and C_n equals C_L ($= \hat{c}_{n+1}x_{n+1}$), the values of $x_1, \ldots, x_n, R_0, \ldots, R_{n-1}$, and $C_0, \ldots, C_{n-1}$ can be found in linear time by applying the idea above, repeatedly. Hence, the modified system can be solved in linear time as summarized in the function $SOLVE(\ )$ in Fig. 5.

In $SOLVE(\ )$, Step 1 follows from (8) with $i = n$ and that $C_L = \hat{c}_{n+1}x_{n+1}$, Step 4 follows from (6), Step 5 follows from (4), Step 6 follows from (8) with $i = i - 1$, Step 9 follows from Lemma 1, Step 10 follows from (5), and Step 11 follows from (9) with $i = i - 1$.

As mentioned above, the solution of the modified system is also a solution of the original system if and only if the value of R_0 returned by $SOLVE(R_n)$ equals $\hat{r}_0/x_0$ ($= R_D$). To find the value of R_n such that $SOLVE(R_n) = R_D$, Lemma 2 below implies that a binary search can be used. The proof of Lemma 2 is given in the Appendix.

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

```
FUNCTION $SOLVE(R_n)$
/* Given $C_L, \mathcal{B}, \mathcal{W}$ and $\hat{r}_i, \hat{c}_i, f_i \forall i$,
   find $x_1, \ldots, x_n, R_0, \ldots, R_n, C_0, \ldots, C_n$.
   Return $R_0$. */
1.  $C_n := C_L (= \hat{c}_{n+1} x_{n+1})$
2.  for $i := n$ downto 1 do
3.      if $i \in \mathcal{B}$ then {
4.          $x_i := \hat{r}_i / R_i$
5.          $R_{i-1} := \hat{r}_i C_i / (\hat{c}_i x_i^2)$
6.          $C_{i-1} := \hat{c}_i x_i$
7.      }
8.      else { /* $i \in \mathcal{W}$ */
9.          $x_i := \frac{\hat{r}_i \hat{c}_i + \sqrt{(\hat{r}_i \hat{c}_i)^2 + 4 \hat{r}_i \hat{c}_i R_i (C_i + f_i/2)}}{2 \hat{c}_i R_i}$
10.         $R_{i-1} := \hat{r}_i (C_i + f_i/2) / (\hat{c}_i x_i^2)$
11.         $C_{i-1} := C_i + \hat{c}_i x_i + f_i$
12.     }
13. return $R_0$
```

Fig. 5. The function to solve the modified system of equations.

```
ALGORITHM $SBWS$
/* Given $\sigma, \epsilon, R_D, C_L, \mathcal{B}, \mathcal{W}$ and $\hat{r}_i, \hat{c}_i, f_i \forall i$,
   find $x_1, \ldots, x_n$. */
1.  $R :=$ an initial guess to the optimal value $R'_n$
2.  if $SOLVE(R) > R_D$, then
3.      while $SOLVE(R) > R_D$ do $R := R/\sigma$
4.  else while $SOLVE(\sigma R) \le R_D$ do $R := \sigma R$
5.  $R_{low} := R$
6.  $R_{up} := \sigma R$
7.  $R_n := (R_{up} + R_{low})/2$
8.  repeat /* Binary search */
9.      if $SOLVE(R_n) < R_D$ then
10.         $R_{low} := R_n$
11.     else $R_{up} := R_n$
12.     $R_n := (R_{up} + R_{low})/2$
13. until $1/(1+\epsilon) \le SOLVE(R_n)/R_D \le 1 + \epsilon$
```

Fig. 6. The simultaneous buffer and wire sizing algorithm for a line.

Lemma 2: $SOLVE(R_n)$ is a strictly increasing function in R_n.

In the following, let $x_1, \ldots, x_n, R_0, \ldots, R_n, C_0, \ldots, C_n$ be the solution computed by $SOLVE(R_n)$, and let $x'_1, \ldots, x'_n, R'_0, \ldots, R'_n, C'_0, \ldots, C'_n$ be the optimal solution. Let D and D' be the delay corresponding to the solution by $SOLVE(R_n)$ and the optimal solution, respectively. Lemma 3 below gives us a condition to terminate the binary search such that the precision of the solution is within ϵ. The proof of Lemma 3 is given in the Appendix.

Lemma 3: For any $\epsilon > 0$, if $1/(1+\epsilon) \le R'_0/R_0 \le 1+\epsilon$, then $|x_i - x'_i|/x'_i < \epsilon$ for $1 \le i \le n$ and $|D - D'|/D' < \epsilon$.

To find a range to start the binary search, we can first make an initial guess R of R_n. Next, R is repeatedly divided or multiplied by a factor σ until $SOLVE(R) \le R_D < SOLVE(\sigma R)$. Then, the range $[R, \sigma R)$ contains the optimal value R'_n and hence can be used to start the binary search. The algorithm SBWS is summarized in Fig. 6.

According to the result of [5], a good initial guess for the value of R in Step 1 is given by

$$R = \begin{cases} \hat{r}_n / x_n, & \text{if } n \in \mathcal{B} \\ \hat{r}_n \left(C_L + \frac{f_n}{2} \right) / (\hat{c}_n x_n^2) + \hat{r}_n / x_n, & \text{if } n \in \mathcal{W} \end{cases}$$

where $\hat{r} = \Sigma_{i \in \mathcal{W}} \hat{r}_i$, $\hat{c} = \Sigma_{i \in \mathcal{W}} \hat{c}_i$, $f = \Sigma_{i \in \mathcal{W}} f_i$, ρ is root of

$$e^{\rho} R_D \left(C_L + \frac{f}{2} \right) \prod_{i \in \mathcal{B} - \{0, n+1\}} \hat{r}_i \hat{c}_i = \left(\frac{\hat{r}\hat{c}}{\rho^2} \right)^{m+1}$$

and

$$x_n = \begin{cases} \rho^2 \hat{r}_n \left(C_L + \frac{f}{2} \right) \Big/ (\hat{r}\hat{c}), & \text{if } n \in \mathcal{B} \\ \rho \left(C_L + \frac{f}{2} \right) \Big/ \hat{c}, & \text{if } n \in \mathcal{W}. \end{cases}$$

In practice, with $\sigma = 2$ and this initial guess, the number of iterations of dividing and multiplying R to find the range is usually only zero or one. In Section V, we demonstrate that these values of σ and initial R work well in practice.

In the following, we prove that with $\sigma = n$ and a simple initial guess of $R = R_D$, the runtime of SBWS is quadratic. Lemma 4 below tells us how close R_n to the optimal value R'_n should be in order to guarantee that the termination condition in Step 13 of SBWS is satisfied. The proof of Lemma 4 is given in the Appendix.

Lemma 4: For any $0 < \epsilon < 1$, if $1/(1 + \epsilon/3^n) \le R'_n/R_n \le 1 + \epsilon/3^n$, then $1/(1+\epsilon) < R'_0/R_0 < 1 + \epsilon$.

So by Lemmas 3 and 4, if $1/(1+\epsilon/3^n) \le R'_n/R_n \le 1 + \epsilon/3^n$, then $|x'_i - x_i|/x_i < \epsilon$ for $1 \le i \le n$ and $|D - D'|/D' < \epsilon$. When we start the binary search, $1/\sigma \le R'_n/R_n \le \sigma$ (where $\sigma = n$ here). So the number of iterations of binary search (i.e., Steps 8–13 of SBWS) to guarantee $1/(1+\epsilon/3^n) \le R'_n/R_n \le 1 + \epsilon/3^n$ is at most $\log_2((\sigma - 1)3^n/\epsilon) = O(n + \log(1/\epsilon))$.

Lemma 5 bounds the number of iterations to find the starting range (i.e., Steps 2–4 of SBWS). Its proof is given in the Appendix.

Lemma 5: With $\sigma = n$ and the initial guess of $R = R_D$, the number of iterations to find the starting range is $O(n)$.

Since it takes $O(n)$ iterations to find the starting range, $O(n + \log(1/\epsilon))$ iterations for the binary search, and each iteration takes $O(n)$ time, we have the following theorem.

Theorem 1: For an interconnect with n components and for any $\epsilon > 0$, the algorithm SBWS solves the simultaneous buffer and wire sizing problem in $O(n^2 + n \log(1/\epsilon))$ time and $O(1)$ memory for computation with precision ϵ (i.e., the solution by SBWS $x_1, \ldots, x_n$, D and the optimal solution $x'_1, \ldots, x'_n$, D' satisfy $|x_i - x'_i|/x'_i < \epsilon$ for all i and $|D - D'|/D' < \epsilon$).

IV. Extensions to Handle Interconnect Trees

So far we have considered interconnects with a line topology. As most interconnects in a circuit have a line topology, SBWS can be applied to them directly. However, there are some interconnects with a tree topology. In this section, we describe how SBWS can be extended to handle interconnects with a tree topology.

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

```
ALGORITHM SBWS-T
1. For each component $i$ in the tree, let $\mu_i$ be the sum
   of all sink weights at the downstream of component $i$.
2. For each component $i$ in the tree,
   scale $\widehat{r}_i$ by the factor $\mu_i$ (i.e., $\widehat{r}_i := \mu_i \widehat{r}_i$).
3. Perform a bottom-up traversal of the tree
   to compute $C_i$ for each component $i$.
4. Perform a top-down traversal of the tree:
   For each tree edge,
       first find the upstream resistance of the edge,
       then optimally size the edge using $SBWS$.
5. Repeat Step 3–4 until no improvement.
```

Fig. 7. The simultaneous buffer and wire sizing algorithm for a tree.

For interconnect trees, minimizing maximum sink delay and minimizing total area subject to sink delay bounds are the most commonly used objectives. Chen *et al.* [4] showed that both objectives can be reduced by the Lagrangian relaxation technique to a sequence of subproblems minimizing a weighted sum of the sink delays. In other words, by solving the problem of minimizing weighted sink delay, we also solve the problems of minimizing maximum sink delay and minimizing total area subject to sink delay bounds as well. So we consider the problem of minimizing weighted sink delay in the following.

To minimize a weighted sum of the sink delays of an interconnect tree, a similar technique as in [4] can be used. The basic idea is to iteratively optimize the tree edges one at a time. At each time an edge is manipulated, we keep all the other edges fixed and apply SBWS to that edge. The corresponding algorithm SBWS-T is given in Fig. 7. Detailed explanations of the steps can be found in [4].

It is easy to see that the weighted sum of the sink delays is a posynomial. So the problem has a unique global minimum. As a result, the algorithm SBWS-T which greedily sizes each edge iteratively always converges to the global minimum. In practice, a few iterations are usually enough for SBWS-T to converge. Note that since the downstream capacitance and upstream resistance of each edge are computed incrementally by a bottom-up traversal in Step 3 and a top-down traversal in Step 4 respectively, each iteration of SBWS-T takes only linear time. Hence, SBWS-T is very efficient in practice. Its efficiency is demonstrated in Section V.

V. Experimental Results and Concluding Remarks

In this section, we show that the algorithms SBWS and SBWS-T are extremely efficient in practice. We have implemented these algorithms in the C Language. We run them on a PC with a 200 MHz Pentium Pro processor. The precision parameter ϵ is set to 0.1%. We use the parameters for the 0.18 μm technology listed in [9].

First, we investigate the runtime of SBWS with respect to the number of components n. Different values of n ranging from 1000–10 000 are used. For each value of n, 100 problem instances are generated randomly. The average CPU time and the average number of calls to the function $SOLVE(\ )$ are reported in Table I. The CPU time is plotted as a function of n in Fig. 8 below.

TABLE I
The Average CPU Time and the Average Number of Calls to the Function $SOLVE(\ )$ for the Algorithm SBWS

n	CPU(s)	# calls
1000	0.025	11.9
2000	0.052	12.1
3000	0.077	12.1
4000	0.102	11.9
5000	0.126	12.0
6000	0.151	11.9
7000	0.176	11.9
8000	0.207	12.1
9000	0.231	12.0
10000	0.255	12.0

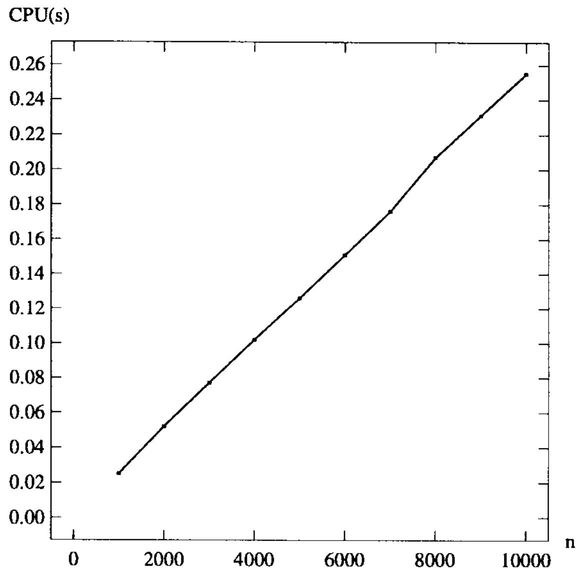

Fig. 8. The CPU time of the algorithm SBWS versus the number of components n.

As the table shows, SBWS is extremely fast in practice. Even for an interconnect of 10 000 components, the CPU time is only 0.255 s. Moreover, we observe that the number of calls to the function $SOLVE(\ )$ is around 12 for all cases. Therefore, as it is clearly demonstrated in Fig. 8, the runtime is linear in practice.

Next, we compare the runtime of SBWS-T with that of GWSA for simultaneous continuous buffer and wire sizing [1]. GWSA is the most efficient algorithm for minimizing weighted sink delay reported in the literature. GWSA can also handle bounds on buffer size and wire width. For the comparison below, we are using a version of GWSA which ignore the bounds on buffer size and wire width.

It has been proved in [6] that for wire-sizing alone, GWSA runs in time linear to the number of wire segments. However, we observe that it is no longer the case when buffer sizing is considered as well. This point is clearly demonstrated by the experiment on a wire below. We divide a 20 000-μm-long wire into 200 segments and we insert m buffers into it, where m ranges from 0 to 10. Then we size it with both SBWS and GWSA. The CPU time versus the number of buffers are plotted in Fig. 9. As shown in Fig. 9, the runtime of SBWS is independent of the number of buffers. However, the runtime

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

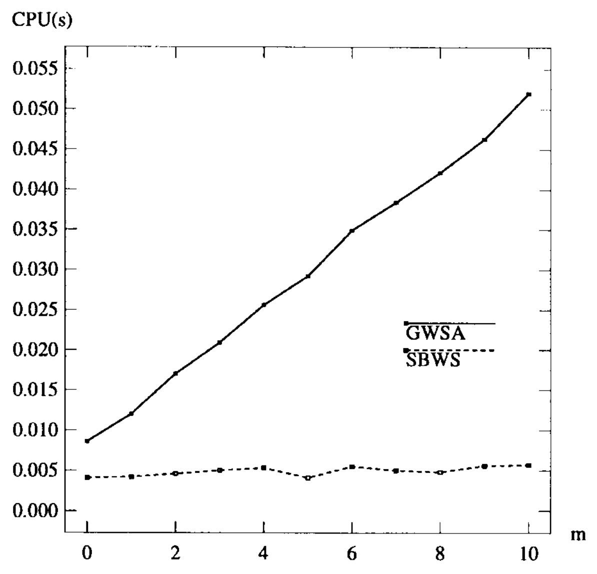

Fig. 9. The CPU time of the algorithms SBWS and GWSA versus the number of buffers m.

TABLE II
THE CPU TIME OF GWSA AND SBWS-T ON INTERCONNECT TREES

Test Case	# sizable components	# sinks	# buffers	CPU Time (s)		
				GWSA	*SBWS-T*	ratio
t1	189	3	4	0.031	0.013	2.38
t2	247	6	7	0.035	0.019	1.84
t3	287	5	5	0.038	0.021	1.81
t4	299	4	7	0.066	0.038	1.74
t5	387	9	9	0.071	0.044	1.61
t6	748	15	17	0.193	0.108	1.79

of GWSA is proportional to the number of buffers in the wire. Also notice that the runtime of SBWS is always better than GWSA, even when there is no buffer.

In Table II, the runtime of SBWS-T and GWSA on six interconnect trees with 4–17 buffers are reported. The length of the tree edges range from 200–12 000 μm. Most edges have zero or one buffer inserted. As shown in Table II, SBWS-T is about twice as fast as GWSA. For more advanced technology, more buffers will be inserted in each edge, and hence larger advantage on the runtime of SBWS-T over GWSA is expected.

We propose the following two directions for future research. First, we would like to see if a tighter analysis will give a better bound on the runtime of SBWS. In Theorem 1, a quadratic runtime is proved. However, the experimental results suggest that the actual runtime of SBWS is close to linear. We conjecture that a much better theoretical bound on the runtime of SBWS is possible. Second, we would like to investigate how our idea can be extended to handle bounds on wire width and buffer size.

APPENDIX
PROOFS OF LEMMAS

In this Appendix, let $x_1, \ldots, x_n, R_0, \ldots, R_n, C_0, \ldots, C_n$ be the solution by $SOLVE(R_n)$, and let $x'_1, \ldots, x'_n, R'_0, \ldots, R'_n, C'_0, \ldots, C'_n$ be the solution by $SOLVE(R'_n)$. For all i, let

$$\begin{aligned}\alpha_i &= R'_i/R_i\\ \beta_i &= \begin{cases} C_i/C'_i & \text{if } i \in \mathcal{B}\\ \left(C_i + \frac{f_i}{2}\right)\Big/\left(C'_i + \frac{f_i}{2}\right) & \text{if } i \in \mathcal{W}\end{cases}\\ \gamma_i &= x_i/x'_i.\end{aligned}$$

Intuitively, $1/\alpha_i$'s, β_i's and γ_i's are the ratios of the upstream resistances, the downstream capacitances and the component sizes of the solutions corresponding to two different values of R_n.

We first introduce Lemmas 6–9 which will be used in proving Lemmas 2–5. Lemmas 6 and 7 below give bounds on γ_i, α_{i-1}, and β_{i-1} based on the values of α_i and β_i for buffers and wire segments, respectively.

Lemma 6: For $1 \leq i \leq n$, if $i \in \mathcal{B}$, then $\gamma_i = \alpha_i$, $\alpha_{i-1} = \gamma_i^2/\beta_i$ and $\beta_{i-1} = \gamma_i$.

Proof: Consider the function $SOLVE(\ )$.

- By Step 4

$$x'_i = \frac{\hat{r}_i}{R'_i} = \frac{\hat{r}_i}{\alpha_i R_i} = \frac{x_i}{\alpha_i}.$$

Therefore, $\gamma_i = \alpha_i$.

- By Step 5

$$R'_{i-1} = \frac{\hat{r}_i C'_i}{\hat{c}_i x_i'^2} = \frac{\hat{r}_i C_i \gamma_i^2}{\hat{c}_i x_i^2 \beta_i} = \frac{\gamma_i^2}{\beta_i} R_{i-1}.$$

Therefore, $\alpha_{i-1} = \gamma_i^2/\beta_i$.

- By Step 6

$$C'_{i-1} = \hat{c}_i x'_i = \frac{\hat{c}_i x_i}{\gamma_i} = \frac{C_{i-1}}{\gamma_i}.$$

Therefore, $\beta_{i-1} = \gamma_i$.

■

Lemma 7: For $1 \leq i \leq n$, if $i \in \mathcal{W}$ and $1 \leq \beta_i < \alpha_i$, then $\sqrt{\alpha_i\beta_i} < \gamma_i < \alpha_i$, $\alpha_{i-1} = \gamma_i^2/\beta_i$, and $1 < \beta_{i-1} < \gamma_i$.

Proof: Consider the function $SOLVE(\ )$.

- By Step 9

$$\begin{aligned} x'_i &= \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_i R'_i\left(C'_i + \frac{f_i}{2}\right)}}{2\hat{c}_i R'_i}\\ &= \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_i\alpha_i R_i\left(C_i + \frac{f_i}{2}\right)\Big/\beta_i}}{2\hat{c}_i\alpha_i R_i}\\ &< \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_i R_i\left(C_i + \frac{f_i}{2}\right)}}{2\hat{c}_i R_i\alpha_i\sqrt{\beta_i/\alpha_i}}\\ &\qquad \text{as } \alpha_i/\beta_i > 1\\ &= x_i/\sqrt{\alpha_i\beta_i}.\end{aligned}$$

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

Therefore, $\sqrt{\alpha_i\beta_i} < \gamma_i$

$$x_i' = \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_iR_i\left(C_i + \frac{f_i}{2}\right)\alpha_i/\beta_i}}{2\hat{c}_i\alpha_iR_i}$$
$$> \frac{\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_iR_i\left(C_i + \frac{f_i}{2}\right)}}{2\hat{c}_i\alpha_iR_i} \quad \text{as } \alpha_i/\beta_i > 1$$
$$= x_i/\alpha_i.$$

Therefore, $\gamma_i < \alpha_i$.

- By Step 10

$$R_{i-1}' = \frac{\hat{r}_i\left(C_i' + \frac{f_i}{2}\right)}{\hat{c}_ix_i'^2} = \frac{\hat{r}_i\left(C_i + \frac{f_i}{2}\right)\gamma_i^2}{\hat{c}_ix_i^2\beta_i} = \frac{\gamma_i^2}{\beta_i}R_{i-1}.$$

Therefore, $\alpha_{i-1} = \gamma_i^2/\beta_i$.

- By Step 11

$$C_{i-1}' + \frac{f_{i-1}}{2} = C_i' + \hat{c}_ix_i' + f_i + \frac{f_{i-1}}{2}$$
$$= \frac{C_i + \frac{f_i}{2}}{\beta_i} + \frac{\hat{c}_ix_i}{\gamma_i} + \frac{f_i}{2} + \frac{f_{i-1}}{2}$$
$$< C_i + \frac{f_i}{2} + \hat{c}_ix_i + \frac{f_i}{2} + \frac{f_{i-1}}{2} \quad \text{as } \beta_i \geq 1 \text{ and } \gamma_i > 1$$
$$= C_{i-1} + \frac{f_{i-1}}{2}.$$

Therefore, $1 < \beta_{i-1}$.

$$C_{i-1}' + \frac{f_{i-1}}{2} = \frac{C_i + \frac{f_i}{2}}{\beta_i} + \frac{\hat{c}_ix_i}{\gamma_i} + \frac{f_i}{2} + \frac{f_{i-1}}{2}$$
$$> \frac{C_i + \frac{f_i}{2} + \hat{c}_ix_i + \frac{f_i}{2} + \frac{f_{i-1}}{2}}{\max(\beta_i, \gamma_i)} \quad \text{as } \beta_i \geq 1 \text{ and } \gamma_i > 1$$
$$= \frac{C_{i-1} + \frac{f_{i-1}}{2}}{\max(\beta_i, \gamma_i)}$$
$$= \frac{C_{i-1} + \frac{f_{i-1}}{2}}{\gamma_i} \quad \text{as } \beta_i < \sqrt{\alpha_i\beta_i} < \gamma_i.$$

Therefore, $\beta_{i-1} < \gamma_i$.

■

Lemmas 8 and 9 below combine the results of Lemmas 6 and 7 so that for all i, the values of β_i, γ_i, and α_{i-1} are bounded based on the value of α_i.

Lemma 8: If $\alpha_n > 1$, then $1 \leq \beta_i < \alpha_i$ for $0 \leq i \leq n$.

Proof: It can be proved by induction on i. Note that $\beta_n = 1$ and it is given that $\alpha_n > 1$. So $1 \leq \beta_n < \alpha_n$.

Assume that $1 \leq \beta_i < \alpha_i$ for some i between 1 and n.

Case 1) $i \in \mathcal{B}$.

By Lemma 6, $\beta_{i-1} = \gamma_i = \alpha_i > 1$ and $\alpha_{i-1} = \gamma_i^2/\beta_i = \alpha_i^2/\beta_i > \alpha_i = \beta_{i-1}$.

Case 2) $i \in \mathcal{W}$.

By Lemma 7, $\beta_{i-1} > 1$ and $\alpha_{i-1} = \gamma_i^2/\beta_i > \alpha_i\beta_i/\beta_i = \alpha_i > \gamma_i > \beta_{i-1}$.

So $1 \leq \beta_{i-1} < \alpha_{i-1}$ for both cases. ■

Lemma 9: If $\alpha_n > 1$, then $1 < \gamma_i \leq \alpha_i$ and $\alpha_i < \alpha_{i-1} < \alpha_i^2$ for $1 \leq i \leq n$.

Proof:

Case 1) $i \in \mathcal{B}$.

By Lemma 6, $\gamma_i = \alpha_i$. By Lemma 8, $1 < \alpha_i = \gamma_i$.

By Lemma 6, $\alpha_{i-1} = \gamma_i^2/\beta_i = \alpha_i^2/\beta_i$. Hence by Lemma 8, $\alpha_i < \alpha_{i-1} < \alpha_i^2$.

Case 2) $i \in \mathcal{W}$.

By Lemmas 7 and 8, $1 < \sqrt{\alpha_i\beta_i} < \gamma_i < \alpha_i$.

By Lemmas 7 and 8, $\alpha_{i-1} = \gamma_i^2/\beta_i > \alpha_i\beta_i/\beta_i = \alpha_i$ and $\alpha_{i-1} = \gamma_i^2/\beta_i < \alpha_i^2/\beta_i \leq \alpha_i^2$.

So $1 < \gamma_i \leq \alpha_i$ and $\alpha_i < \alpha_{i-1} < \alpha_i^2$ for both cases. ■

Proof of Lemma 2: Suppose $R_n' > R_n$. Then $\alpha_n > 1$. So by Lemma 9, $1 < \alpha_n < \cdots < \alpha_1 < \alpha_0$. In particular, $\alpha_0 > 1$. So $R_0' > R_0$. In other words, $SOLVE(R_n)$ is a strictly increasing function in R_n. ■

Proof of Lemma 3: If $R_n' > R_n$, then $\alpha_n > 1$. So by Lemma 9, $1 < \alpha_n < \cdots < \alpha_1 < \alpha_0$ and $1 < \gamma_i \leq \alpha_i$ for $1 \leq i \leq n$, and by Lemma 8, $1 \leq \beta_i < \alpha_i$ for $0 \leq i \leq n$. It is given that $\alpha_0 = R_0'/R_0 \leq 1 + \epsilon$. Therefore, $1 < \gamma_i < 1 + \epsilon$ and $1 \leq \beta_i < 1 + \epsilon$ for all i. $1 < \gamma_i < 1 + \epsilon$ implies $0 < (x_i - x_i')/x_i' < \epsilon$ for $1 \leq i \leq n$. The delay expression in (3) can be rewritten as

$$D = \sum_{i \in \mathcal{B} - \{n+1\}} \frac{\hat{r}_i}{x_i}C_i + \sum_{i \in \mathcal{W}} \frac{\hat{r}_i}{x_i}\left(C_i + \frac{f_i}{2}\right) + \text{constant terms.}$$

By the definitions of γ_i and β_i

$$D' = \sum_{i \in \mathcal{B} - \{n+1\}} \frac{\hat{r}_i}{x_i'}C_i' + \sum_{i \in \mathcal{W}} \frac{\hat{r}_i}{x_i'}\left(C_i' + \frac{f_i}{2}\right) + \text{constant terms}$$
$$= \sum_{i \in \mathcal{B} - \{n+1\}} \frac{\beta_i}{\gamma_i}\frac{\hat{r}_i}{x_i}C_i + \sum_{i \in \mathcal{W}} \frac{\beta_i}{\gamma_i}\frac{\hat{r}_i}{x_i}\left(C_i + \frac{f_i}{2}\right) + \text{constant terms.}$$

Since $1 < \gamma_i < 1 + \epsilon$ and $1 \leq \beta_i < 1 + \epsilon$ for all i, $1/(1 + \epsilon) < \beta_i/\gamma_i < 1 + \epsilon$ for all i. Hence, $1/(1 + \epsilon) < D/D' < 1 + \epsilon$, which implies, $|D - D'|/D' < \epsilon$.

If $R_n' < R_n$, using $1/(1 + \epsilon) \leq R_0'/R_0$, we can prove similarly (but with the roles of the solutions by $SOLVE(R_n)$ and $SOLVE(R_n')$ exchanged) that $-\epsilon < (x_i - x_i')/x_i' < 0$ for $1 \leq i \leq n$ and $|D - D'|/D' < \epsilon$. ■

Proof of Lemma 4: If $R_n' > R_n$, then $\alpha_n > 1$. So by Lemma 9, $\alpha_{i-1} < \alpha_i^2$ for $1 \leq i \leq n$. As $\alpha_n \leq 1 + \epsilon/3^n$, $\alpha_{n-1} \leq (1 + \epsilon/3^n)^2 = 1 + 2\epsilon/3^n + (\epsilon/3^n)^2 < 1 +$

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.

$3\epsilon/3^n = 1 + \epsilon/3^{n-1}$. We can apply the idea inductively to show that $\alpha_0 < 1 + \epsilon$. Therefore, together with Lemma 2, $1 < R'_0/R_0 < 1 + \epsilon$.

If $R'_n < R_n$, using $1/(1+\epsilon/3^n) \le R'_n/R_n$, we can prove similarly that $1/(1+\epsilon) < R'_0/R_0 < 1$. ■

Proof of Lemma 5: First, we want to upper bound the optimal delay D'. When $x_1 = \cdots = x_n = 1$, the resistance and capacitance of all components are constant. Since the Elmore delay is a sum of $O(n^2)$ terms such that each term is a product of the resistance of some component and the capacitance of some other component, the delay when $x_1 = \cdots = x_n = 1$ is upper bounded by $O(n^2)$. So the optimal delay $D' \le$ (delay when $x_1 = \cdots = x_n = 1) \le O(n^2)$.

- To prove $1/n^{O(n)} \le R'_n/R_D$:

If $i \in \mathcal{B}$, then by Steps 4 and 6 of $SOLVE(\ )$, $R'_iC'_{i-1} = \hat{r}_i\hat{c}_i$. If $i \in \mathcal{W}$, then

$$\begin{aligned} R'_iC'_{i-1} &= R'_i(C'_i + \hat{c}_ix'_i + f_i) \\ &\quad \text{by Step 11 of } SOLVE(\ ) \\ &> R'_i\hat{c}_ix'_i \\ &= \left(\hat{r}_i\hat{c}_i + \sqrt{(\hat{r}_i\hat{c}_i)^2 + 4\hat{r}_i\hat{c}_iR'_i(C'_i + f_i/2)}\right)/2 \\ &\quad \text{by Step 9 of } SOLVE(\ ) \\ &> \hat{r}_i\hat{c}_i. \end{aligned}$$

So $\hat{c}_i\hat{r}_i \le C'_{i-1}R'_i$ for all i. Moreover, by the definition of Elmore delay, $R'_iC'_i < D'$ for all i. Hence

$$\begin{aligned} R_D\prod_{i=1}^{n}\hat{c}_i\hat{r}_i &= R'_0 \cdot \hat{c}_1\hat{r}_1 \cdot \hat{c}_2\hat{r}_2 \cdots \hat{c}_n\hat{r}_n \\ &\le R'_0 \cdot C'_0R'_1 \cdot C'_1R'_2 \cdots C'_{n-1}R'_n \\ &= R'_0C'_0 \cdot R'_1C'_1 \cdots R'_{n-1}C'_{n-1} \cdot R'_n \\ &< D' \cdot D' \cdots D' \cdot R'_n \\ &= (D')^nR'_n. \end{aligned}$$

Therefore $R'_n/R_D \ge \Pi_{i=1}^n \hat{c}_i\hat{r}_i/O(n^2)^n \ge 1/n^{O(n)}$.

- To prove $R'_n/R_D \le O(n^2)$:

Since $R'_nC_L \le D'$, $R'_n/R_D \le D'/(R_DC_L) \le O(n^2)$.

Hence, $1/n^{O(n)} \le R'_n/R_D \le O(n^2)$. To bring the initial value R_D close to the optimal value R'_n, the number of iterations is bounded by $\log_\sigma n^{O(n)} + \log_\sigma O(n^2) = O(n)$ (where $\sigma = n$, here). ■

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] C.-P. Chen, Y.-W. Chang, and D. F. Wong, "Fast performance driven optimization for buffered clock trees based on Lagrangian relaxation," in *Proc. ACM/IEEE Design Automation Conf.*, 1996, pp. 405–408.

[2] C.-P. Chen and D. F. Wong, "A fast algorithm for optimal wire-sizing under Elmore delay model," in *Proc. IEEE ISCAS*, 1996, vol. 4, pp. 412–415.

[3] ——, "Optimal wire-sizing function with fringing capacitance consideration," in *Proc. ACM/IEEE Design Automation Conf.*, 1997, pp. 604–607.

[4] C.-P. Chen, H. Zhou, and D. F. Wong, "Optimal nonuniform wire-sizing under the Elmore delay model," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1996, pp. 38–43.

[5] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," in *Proc. Int. Symp. on Physical Design*, 1997, pp. 192–197.

[6] ——, "Greedy wire-sizing is linear time," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 398–405, Apr. 1999.

[7] ——, "A quadratic programming approach to simultaneous buffer insertion/sizing and wire sizing," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 787–798, June 1999.

[8] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. Design Automat. Electron. Syst.*, vol. 14, Oct. 1996.

[9] J. Cong, L. He, K.-Y. Khoo, C.-K. Koh, and Z. Pan, "Interconnect design for deep submicron IC's," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1997, pp. 478–485.

[10] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *INTEGRATION, the VLSI J.*, vol. 21, pp. 1–94, 1996.

[11] J. Cong, C.-K. Koh, and K.-S. Leung, "Simultaneous buffer and wire sizing for performance and power optimization," in *Proc. Int. Symp. on Low Power Electronics and Design*, Aug. 1996, pp. 271–276.

[12] J. Cong and K.-S. Leung, "Optimal wiresizing under the distributed Elmore delay model," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1993, pp. 634–639.

[13] R. J. Duffin, E. L. Peterson, and C. Zener, *Geometric Programming—Theory and Application*. New York: Wiley, 1967.

[14] 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.

[15] J. P. Fishburn, "Shaping a VLSI wire to minimize Elmore delay," in *Proc. European Design and Test Conf.*, 1997.

[16] 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, pp. 437–447, Mar. 1996.

[17] N. Menezes, R. Baldick, and L. T. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1995, pp. 144–151.

[18] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387–391.

[19] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, 1997.

[20] N. H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, 2nd ed. Reading, MA: Addison-Wesley, 1993.

Chris C. N. Chu, for a photograph and biography, see p. 405 of the April 1999 issue of this TRANSACTIONS.

D. F. Wong (M'88), for a photograph and biography, see p. 374 of the April 1999 issue of this TRANSACTIONS.

Authorized licensed use limited to: Iowa State University. Downloaded on April 15,2010 at 14:42:44 UTC from IEEE Xplore. Restrictions apply.