A Polynomial Time Optimal Algorithm for Simultaneous Buffer and Wire Sizing*

Chris C. N. Chu and D. F. Wong
cnchu@cs.utexas.edu and wong@cs.utexas.edu
Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, U.S.A.

Abstract

An interconnect joining a source and a sink is divided into fixed-length uniform-width wire segments, and some adjacent segments have buffers in between. The problem we considered is to simultaneously size the buffers and the segments so that the Elmore delay from the source to the sink is minimized. Previously, no polynomial time algorithm for the problem has been reported in literature. In this paper, we present a polynomial time algorithm *SBWS* for the simultaneous buffer and wire sizing problem. *SBWS* is an iterative algorithm with guaranteed convergence to the optimal solution. It runs in quadratic time and uses constant memory for computation. Also, experimental results show that *SBWS* is extremely efficient in practice. For example, for an interconnect of 10000 segments and buffers, the CPU time is only 0.127 second.

1 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 down to $0.25\mu m$ in advance technology. Interconnect delay has become the dominating factor in determining system performance. In many systems designed today, as much as 50% to 70% of clock cycle are consumed by interconnect delay [8]. It is predicted in [11] that the feature size will be reduced to $0.18\mu m$ by 1999 and $0.13\mu m$ by 2002. So we expect the significance of interconnect delay will further increase in the near future.

Both buffer sizing and wire sizing have been shown to be effective techniques to reduce interconnect delay and many works have been done during the past few years. For example, [2, 3, 4, 10, 14] are various results on wire sizing alone. [16] applies the sequential quadratic programming approach to simultaneous gate and wire sizing. This algorithm is comparatively slow as it has to solve a sequence of quadratic programming subproblems. Also, no bound on the run time of the algorithm is reported. [15] gives an algorithm for simultaneous buffer insertion, buffer sizing and wire sizing based on dynamic programming. However, their algorithm runs in pseudo-polynomial time and requires a substantial amount of memory. [1, 7, 9] give greedy algorithms for simultaneous transistor/buffer and wire sizing. These algorithms are shown to be very efficient in practice. However, no bounds on the run time of them are known. [5] considers buffer insertion, buffer sizing and wire sizing simultaneously and a closed form optimal solution is obtained. However, in that paper, only wire area capacitance is considered. Wire fringing capacitance, which will become more and more significant as feature size decreases, is ignored. Taking wire fringing capacitance into account significantly complicates the problem and [5] can only give an approximate solution. [6] shows 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. However, if buffer sizing is considered also, only a brute-force enumeration of the buffer sizes is proposed. See [8] for a comprehensive survey on previous works.

In this paper, we consider the problem of minimizing interconnect delay by simultaneously sizing buffers and wire segments. Basically, an interconnect joining a source and a sink is divided into some fixed-length 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. In particular, both wire area capacitance and wire fringing capacitance are taken into account, and an approach completely different from that in [5] is required here. The details of the problem formulation are discussed in Section 2.

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.

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658288 and by a grant from the Intel Corporation.

- We prove that for an interconnect wire consisting of n buffers and segments, *SBWS* runs in $O(n^2 + n \log \frac{1}{\epsilon})$ time, where ϵ specifies the precision of computation (see Theorem 1). Since $\log \frac{1}{\epsilon}$ is bounded by the number of bits in the input, the total run time 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 demonstrate experimentally that *SBWS* is also extremely efficient in practice. For example, for an interconnect of 10000 segments and buffers, the CPU time is only 0.127 second. Besides, we observe that *SBWS* runs in linear time in practice.

The rest of the paper is organized as follows. In Section 2, we present the formulation of the simultaneous buffer and wire sizing problem. In Section 3, the algorithm *SBWS*, its optimality proof and its run time analysis are presented. In Section 4, some experimental results to show the efficiency of *SBWS* are presented. In Section 5, we discuss some extensions of our results.

2 Problem Formulation

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 i-th 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 $x_1, \ldots, x_n$. See Figure 1 for an illustration.

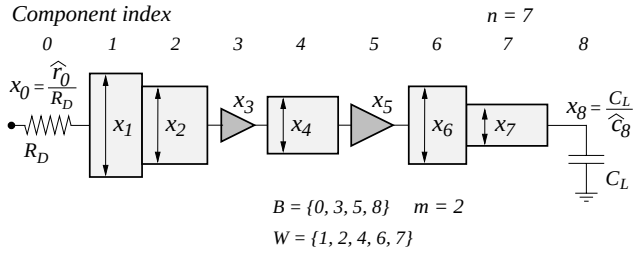

Figure 1: *The simultaneous buffer and wire sizing problem. Given the driver resistance R_D, the load capacitance C_L, the number of components n, the set of component indexes of buffers $\mathcal{B}$, and the set of component indexes of wire segments $\mathcal{W}$, the objective is to find the optimal wire widths and buffer sizes $x_1, \ldots, x_n$ such that the delay from the source to the sink is minimized.*

In general, the source and the sink can be anything. However, in order to simplify the notations, we will treat them as buffers of fixed size in this paper. Let the source be called the 0-th 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 $1 \le j \le m$, let b_j be the component index of the j-th sizable buffer. Let $b_0 = 0$ 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. $\mathcal{W} = \{0, 1, \ldots, n+1\} - \mathcal{B}$.

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 Figure 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 $C_L = \hat{c}_{n+1} x_{n+1}$.

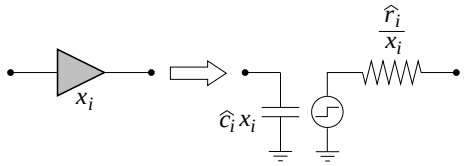

Figure 2: *The model of a buffer as a switch-level RC circuit.*

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 Figure 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.

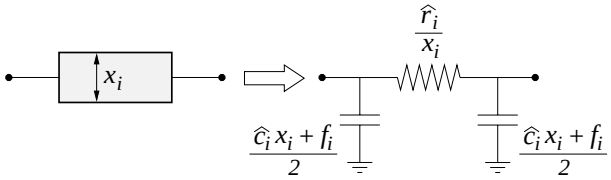

Figure 3: *The model of a wire segment as a π-type RC circuit.*

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

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

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

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 after component i (up to the next buffer).

See Figure 4 for an illustration. Let the *upstream capacitances* of component i be R_{i-1}. Let the *downstream capacitances* of component i be C_i if $i \in \mathcal{B}$, or $C_i + f_i/2$ if $i \in \mathcal{W}$.

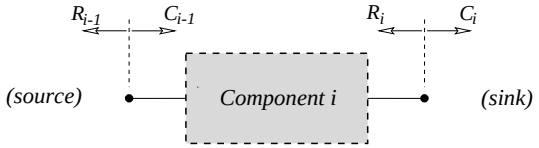

Figure 4: *Illustration of R_i and C_i.*

In this paper, the widely used Elmore delay model [13] 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

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

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

3 The Algorithm *SBWS*

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

$$R_i = \widehat{r}_i / x_i \quad \text{if } i \in \mathcal{B} \quad (4)$$
$$R_i = R_{i-1} + \widehat{r}_i / x_i \quad \text{if } i \in \mathcal{W} \quad (5)$$

$$C_i = \widehat{c}_{i+1} x_{i+1} \quad \text{if } i+1 \in \mathcal{B} \quad (6)$$
$$C_i = C_{i+1} + \widehat{c}_{i+1} x_{i+1} + f_{i+1} \quad \text{if } i+1 \in \mathcal{W} \quad (7)$$

Next, necessary and sufficient conditions for optimality will be derived. For $1 \le i \le n$, if $i \in \mathcal{B}$, then we can write D in (3) in term of x_i as

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

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

$$\widehat{c}_i R_{i-1} x_i^2 = \widehat{r}_i C_i \quad (8)$$

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

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

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

$$\widehat{c}_i R_{i-1} x_i^2 = \widehat{r}_i (C_i + f_i/2) \quad (9)$$

Note that D is a posynomial [12] in $x_1, \ldots, x_n$. It is well known that under a variable transformation, a posynomial is equivalent to a convex function. 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, (8) and (9) are necessary and sufficient conditions for optimality.

As a result, finding the optimal solution to the problem is equivalent to solving (8) and (9) for $x_1, \ldots, x_n$, where $R_0, \ldots, R_n, C_0, \ldots, C_n$ satisfy (4), (5), (6) and (7). Instead of solving the system of equations (4), (5), (6), (7), (8) and (9) directly, we consider a modified system obtained by ignoring the equation $R_0 = \widehat{r}_0 / x_0$ (one of the equations in (4) when $i = 0$) and adding an extra equation to fix the value of R_n. If the resulting R_0 equals $\widehat{r}_0 / x_0$ ($= R_D$ 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.

We will show in the following how to solve the modified system of equations in linear time. First of all, we have to prove the lemma below which relates x_i, R_i and C_i for any wire segment i.

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

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

Proof: Eliminating R_{i-1} from (5) and (9), we have $\widehat{c}_i (R_i - \widehat{r}_i / x_i) x_i^2 = \widehat{r}_i (C_i + f_i/2)$, or equivalently, $\widehat{c}_i R_i x_i^2 - \widehat{r}_i \widehat{c}_i x_i - \widehat{r}_i (C_i + f_i/2) = 0$. 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 (4) (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 ($= \widehat{c}_{n+1} x_{n+1}$), 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 in the procedure *SOLVE*() below.

In *SOLVE*(), step 1 follows from (6) with $i = n$ and that $C_L = \widehat{c}_{n+1} x_{n+1}$, step 4 follows from (4), step 5 follows from (8), step 6 follows from (6) with $i = i - 1$, step 9 follows from Lemma 1, step 10 follows from (9), and step 11 follows from (7) with $i = i - 1$.

As mentioned above, in order that the solution of the modified system is also a solution of the original system, the value of R_0 computed by *SOLVE*(R_n) must equal $\widehat{r}_0 / x_0$ ($= R_D$). For convenience, we define the function

PROCEDURE *SOLVE*(R_n)
(Assume that $C_L, \mathcal{B}, \mathcal{W}$ and $\forall i\ \widehat{r}_i, \widehat{c}_i, f_i$ are given.)
Input: R_n
Output: $x_1, \ldots, x_n, R_0, \ldots, R_n, C_0, \ldots, C_n$

1. $C_n := C_L (= \widehat{c}_{n+1} x_{n+1})$
2. **for** $i := n$ **downto** 1 **do** {
3. **if** $i \in \mathcal{B}$ **then** {
4. $x_i := \widehat{r}_i / R_i$
5. $R_{i-1} := \widehat{r}_i C_i / (\widehat{c}_i x_i^2)$
6. $C_{i-1} := \widehat{c}_i x_i$
7. }
8. **else** { /* $i \in \mathcal{W}$ */
9. $x_i := \dfrac{\widehat{r}_i\widehat{c}_i + \sqrt{(\widehat{r}_i\widehat{c}_i)^2 + 4\widehat{r}_i\widehat{c}_i R_i (C_i + f_i/2)}}{2\widehat{c}_i R_i}$
10. $R_{i-1} := \widehat{r}_i (C_i + f_i/2) / (\widehat{c}_i x_i^2)$
11. $C_{i-1} := C_i + \widehat{c}_i x_i + f_i$
12. }
13. }

$R_0(R_n)$, which simply returns the value of R_0 computed by *SOLVE*(R_n).

FUNCTION $R_0(R_n)$
Input: R_n
Output: R_0 computed by *SOLVE*(R_n)

1. Call *SOLVE*(R_n)
2. **return** R_0

We will show in the following how to find the value of R_n such that $R_0(R_n)$ equals R_D.

Let $x_1, \ldots, x_n, R_0, \ldots, R_n, C_0, \ldots, C_n$ and $x'_1, \ldots, x'_n$, $R'_0, \ldots, R'_n, C'_0, \ldots, C'_n$ be the output of *SOLVE*(R_n) and *SOLVE*(R'_n) respectively. For all i, let

$$\alpha_i = R'_i / R_i \quad (\text{i.e. } R'_i = \alpha_i R_i)$$

$$\beta_i = \begin{cases} C_i / C'_i \ (\text{i.e. } C'_i = C_i/\beta_i) & \text{if } i \in \mathcal{B} \\ (C_i + \frac{f_i}{2}) / (C'_i + \frac{f_i}{2}) & \\ \quad (\text{i.e. } C'_i + \frac{f_i}{2} = (C_i + \frac{f_i}{2})/\beta_i) & \text{if } i \in \mathcal{W} \end{cases}$$

$$\gamma_i = x_i / x'_i \quad (\text{i.e. } x'_i = x_i/\gamma_i)$$

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.

Lemma 2 and Lemma 3 give bounds on the values of γ_i, α_{i-1} and β_{i-1} based on the values of α_i and β_i for $i \in \mathcal{B}$ and $i \in \mathcal{W}$ respectively.

Lemma 2 *For $1 \le i \le n$, 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 procedure *SOLVE*().

- By step 4, $x'_i = \dfrac{\widehat{r}_{i'}}{R'_i} = \dfrac{\widehat{r}_{i'}}{\alpha_i R_i} = \dfrac{x_i}{\alpha_i}$.
  Therefore $\gamma_i = \alpha_i$.
- By step 5, $R'_{i-1} = \dfrac{\widehat{r}_i C'_i}{\widehat{c}_i {x'_i}^2} = \dfrac{\widehat{r}_i C_i {\gamma_i}^2}{\widehat{c}_i x_i^2 \beta_i} = \dfrac{{\gamma_i}^2}{\beta_i} R_{i-1}$.
  Therefore $\alpha_{i-1} = \gamma_i^2/\beta_i$.
- By step 6, $C'_{i-1} = \widehat{c}_i x'_i = \dfrac{\widehat{c}_i x_i}{\gamma_i} = \dfrac{C_{i-1}}{\gamma_i}$.
  Therefore $\beta_{i-1} = \gamma_i$.

□

Lemma 3 *For $1 \le i \le n$, if $i \in \mathcal{W}$ and $1 \le \beta_i < \alpha_i$, 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 procedure *SOLVE*().

- By step 9,

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

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

  Also,

$$\begin{aligned} x'_i &= \frac{\widehat{r}_i\widehat{c}_i + \sqrt{(\widehat{r}_i\widehat{c}_i)^2 + 4\widehat{r}_i\widehat{c}_i R_i (C_i + \frac{f_i}{2})\alpha_i/\beta_i}}{2\widehat{c}_i \alpha_i R_i} \\ &> \frac{\widehat{r}_i\widehat{c}_i + \sqrt{(\widehat{r}_i\widehat{c}_i)^2 + 4\widehat{r}_i\widehat{c}_i R_i (C_i + \frac{f_i}{2})}}{2\widehat{c}_i \alpha_i R_i} \\ &\qquad \text{as } \alpha_i/\beta_i > 1 \\ &= x_i/\alpha_i \end{aligned}$$

  Therefore $\gamma_i < \alpha_i$.
- By step 10,

$$R'_{i-1} = \frac{\widehat{r}_i (C'_i + \frac{f_i}{2})}{\widehat{c}_i {x'_i}^2} = \frac{\widehat{r}_i (C_i + \frac{f_i}{2}) {\gamma_i}^2}{\widehat{c}_i x_i^2 \beta_i} = \frac{{\gamma_i}^2}{\beta_i} R_{i-1}$$

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

- By step 11,

$$\begin{aligned} C'_{i-1} + \frac{f_{i-1}}{2} &= C'_i + \widehat{c}_i x'_i + f_i + \frac{f_{i-1}}{2} \\ &= \frac{C_i + \frac{f_i}{2}}{\beta_i} + \frac{\widehat{c}_i x_i}{\gamma_i} + \frac{f_i}{2} + \frac{f_{i-1}}{2} \\ &< C_i + \frac{f_i}{2} + \widehat{c}_i x_i + \frac{f_i}{2} + \frac{f_{i-1}}{2} \\ &\quad \text{as } \beta_i \geq 1 \text{ and } \gamma_i > 1 \\ &= C_{i-1} + \frac{f_{i-1}}{2} \end{aligned}$$

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

$$\begin{aligned} C'_{i-1} + \frac{f_{i-1}}{2} &= \frac{C_i + \frac{f_i}{2}}{\beta_i} + \frac{\widehat{c}_i x_i}{\gamma_i} + \frac{f_i}{2} + \frac{f_{i-1}}{2} \\ &> \frac{C_i + \frac{f_i}{2} + \widehat{c}_i x_i + \frac{f_i}{2} + \frac{f_{i-1}}{2}}{\max(\beta_i, \gamma_i)} \\ &\quad \text{as } \beta_i \geq 1 \text{ and } \gamma_i > 1 \\ &= \frac{C_{i-1} + \frac{f_{i-1}}{2}}{\max(\beta_i, \gamma_i)} \\ &= \frac{C_{i-1} + \frac{f_{i-1}}{2}}{\gamma_i} \\ &\quad \text{as } \beta_i < \sqrt{\alpha_i \beta_i} < \gamma_i \end{aligned}$$

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

□

In Lemma 4 and Lemma 5 below, the results of Lemma 2 and Lemma 3 are combined 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 4 *If $\alpha_n > 1$, then $1 \leq \beta_i < \alpha_i$ for $1 \leq i \leq n$.*

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.

Case 1) $i \in \mathcal{B}$:
By Lemma 2, $\beta_{i-1} = \gamma_i = \alpha_i > 1$ 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 3, $\beta_{i-1} > 1$ 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 5 *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 2, $\gamma_i = \alpha_i$. By Lemma 4, $1 < \alpha_i = \gamma_i$. By Lemma 2, $\alpha_{i-1} = \gamma_i^2/\beta_i = \alpha_i^2/\beta_i$. Hence by Lemma 4, $\alpha_i < \alpha_{i-1} < \alpha_i^2$.

Case 2) $i \in \mathcal{W}$:
By Lemma 3 and Lemma 4, $1 < \sqrt{\alpha_i\beta_i} < \gamma_i < \alpha_i$. By Lemma 3 and Lemma 4, $\alpha_{i-1} = \gamma_i^2/\beta_i > \alpha_i\beta_i/\beta_i = \alpha_i$ 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. □

Lemma 6 *$R_0(R_n)$ is a strictly increasing function.*

Proof: Suppose $R'_n > R_n$. Then $\alpha_n > 1$. So by Lemma 5, $1 < \alpha_n < \cdots < \alpha_1 < \alpha_0$. In particular, $\alpha_0 > 1$. So $R'_0 > R_0$. In other words, $R_0(R_n)$ is a strictly increasing function in R_n. □

For the rest of this section, $x_1, \ldots, x_n$, $R_0, \ldots, R_n$, $C_0, \ldots, C_n$ will be viewed as the optimal solution, and $x'_1, \ldots, x'_n$, $R'_0, \ldots, R'_n$, $C'_0, \ldots, C'_n$ as the solution by $SOLVE(R'_n)$ for some R'_n.

Lemma 7 *For any $\epsilon > 0$, if $1/(1+\epsilon) \leq R'_0/R_0 \leq 1+\epsilon$, then $|x'_i - x_i|/x_i < \epsilon$ for $1 \leq i \leq n$.*

Proof: If $R'_n > R_n$, then $\alpha_n > 1$. So by Lemma 5, $1 < \alpha_n < \cdots < \alpha_1 < \alpha_0$ and $1 < \gamma_i \leq \alpha_i$ for $1 \leq i \leq n$. It is given that $\alpha_0 = R'_0/R_0 \leq 1+\epsilon$. Therefore, for $1 \leq i \leq n$, $1 < \gamma_i < 1+\epsilon$, or equivalently, $1/(1+\epsilon) < x'_i/x_i < 1$. This implies $-\epsilon < (x'_i - x_i)/x_i < 0$ for $1 \leq i \leq n$.

If $R'_n < R_n$, using $1/(1+\epsilon) \leq R'_0/R_0$, we can prove similarly (but with the roles of the optimal buffer and wire sizing solution and the output of $SOLVE(R'_n)$ exchanged) that $0 < (x'_i - x_i)/x_i < \epsilon$ for $1 \leq i \leq n$.

Hence the lemma follows. □

To find the value of R_n such that $R_0(R_n) = R_D$, Lemma 6 implies that binary search can be used. Lemma 7 gives us a condition to terminate the binary search such that the precision of the solution is within ϵ. So what is left now is a range to start the binary search. We can find it by first making an initial guess R of R_n. Next, R is repeatedly divided or multiplied by 2 until $SOLVE(R) \leq R_D < SOLVE(2R)$. Then the range $[R, 2R)$ will contain the optimal R_n and hence can be used to start the binary search. The algorithm is summarized below.

A good initial guess for the value of R in step 1 can be obtained by the result of [5]. When there is no fringing capacitance, we can use [5] to find the exact value of the optimal x_n. With fringing capacitance (as in our case), we can use it to obtain a good approximation to x_n, and hence a good approximation to R_n. The formula to

ALGORITHM *SBWS*
Input: $\epsilon, R_D, C_L, \mathcal{B}, \mathcal{W}$ and $\forall i\ \widehat{r}_i, \widehat{c}_i, f_i$
Output: $x_1, \ldots, x_n$
1. $R :=$ an initial guess of the optimal R_n
2. **if** $R_0(R) > R_D$, **then**
3. **while** $R_0(R) > R_D$ **do** $R := R/2$
4. **else while** $R_0(2R) \leq R_D$ **do** $R := 2R$
5. $R_{low} := R$
6. $R_{up} := 2R$
7. $R_n := (R_{up} + R_{low})/2$
8. **repeat** /* Binary search */
9. **if** $R_0(R_n) < R_D$ **then**
10. $R_{low} := R_n$
11. **else** $R_{up} := R_n$
12. $R_n := (R_{up} + R_{low})/2$
13. **until** $1/(1+\epsilon) \leq R_0(R_n)/R_D \leq 1+\epsilon$

find the initial guess are listed below (explanations are omitted due to space limitation).

Let $\widehat{r} = \sum_{i \in \mathcal{W}} \widehat{r}_i$, $\widehat{c} = \sum_{i \in \mathcal{W}} \widehat{c}_i$, and $f = \sum_{i \in \mathcal{W}} f_i$.

Let ρ be the value such that

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

Let $x_n = \begin{cases} \rho^2 \widehat{r}_n (C_L + \frac{f}{2})/(\widehat{r}\widehat{c}) & \text{if } n \in \mathcal{B} \\ \rho (C_L + \frac{f}{2})/\widehat{c} & \text{if } n \in \mathcal{W} \end{cases}$

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

We can show that the number of iterations of dividing and multiplying R to find the range is $O(1)$. In practice, usually only zero or one iteration is needed.

To bound the number of iterations that binary search takes, we need the following lemma.

Lemma 8 *For any $0 < \epsilon < 1$, if $1/(1+\epsilon/3^n) \leq R'_n/R_n \leq 1+\epsilon/3^n$, then $1/(1+\epsilon) < R'_0/R_0 < 1+\epsilon$.*

Proof: If $R'_n > R_n$, then $\alpha_n > 1$. So by Lemma 5, $\alpha_{i-1} < \alpha_i^2$ for $1 \leq i \leq n$. As $\alpha_n \leq 1+\epsilon/3^n$, $\alpha_{n-1} \leq (1+\epsilon/3^n)^2 = 1+2\epsilon/3^n+(\epsilon/3^n)^2 < 1+3\epsilon/3^n = 1+\epsilon/3^{n-1}$. We can apply the idea inductively to show that $\alpha_0 < 1+\epsilon$. Therefore, together with Lemma 6, $1 < R'_0/R_0 < 1+\epsilon$.

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

So by Lemma 7 and Lemma 8, if $1/(1+\epsilon/3^n) \leq R'_n/R_n \leq 1+\epsilon/3^n$, then $|x'_i - x_i|/x_i < \epsilon$ for $1 \leq i \leq n$. The number of iterations for the binary search to guarantee $1/(1+\epsilon/3^n) \leq R'_n/R_n \leq 1+\epsilon/3^n$ is at most $O(\log(3^n/\epsilon)) = O(n + \log\frac{1}{\epsilon})$. Since 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 $O(n^2 + n\log\frac{1}{\epsilon})$ time and $O(1)$ memory for computation with precision ϵ (i.e. the optimal solution $x_1, \ldots, x_n$ and the solution by SBWS $x'_1, \ldots, x'_n$ satisfy $|x'_i - x_i|/x_i < \epsilon$ for all i).*

4 Experimental Results

In this section, we will show that the algorithm *SBWS* is extremely efficient in practice. We have implemented *SBWS* in C. We run it on a IBM PC with a 200 MHz Pentium Pro processor. The precision parameter ϵ is set to 0.1%. Different values for the number of components n ranging from 1000 to 10000 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 procedure *SOLVE*() are reported in Table 1. As the table shows, *SBWS* is extremely fast in practice. Even for an interconnect of 10000 components, the CPU time is only 0.127s.

n	CPU(s)	# calls
1000	0.013	11.9
2000	0.026	12.1
3000	0.039	12.1
4000	0.051	11.9
5000	0.063	12.0
6000	0.076	11.9
7000	0.088	11.9
8000	0.104	12.1
9000	0.115	12.0
10000	0.127	12.0

Table 1: *The average CPU time and the average number of calls to the procedure SOLVE() for the algorithm SBWS.*

Moreover, we observe that the number of calls to the procedure *SOLVE*() is around 12 for all cases. So run time is linear in practice. The CPU time is plotted as a function of n in Figure 5 below.

5 Discussion

Our result can be extended in several ways:

Wire area and power consideration: Our algorithm *SBWS* can be extended easily to minimize a weighted sum of total wire area, power and delay. As the objective in (3) is changed, the optimality conditions (8) and (9) will also be the different. However, it is not difficult to see that the problem can still be solved by the ideas of this paper without much modification. For other objectives like minimizing delay subject to area bound or minimizing area subject to delay bound, we can apply the Lagrangian relaxation technique as in [4] to reduce the problems to a problem of minimizing a weighted sum.

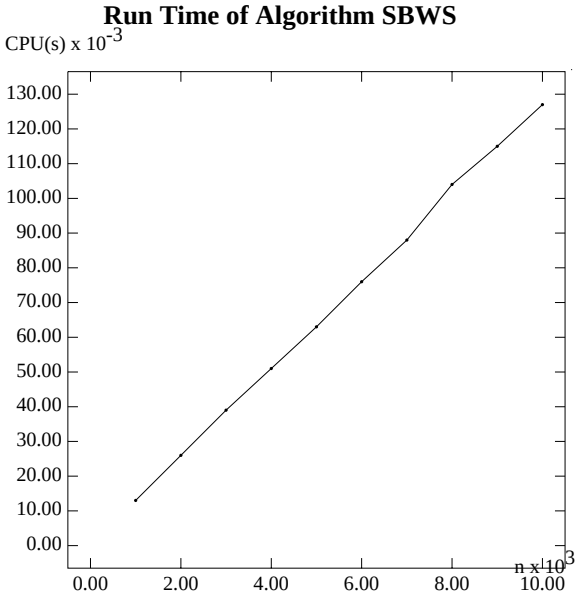

Figure 5: *The average CPU time of the algorithm SBWS.*

Interconnect with tree topology: *SBWS* is designed for interconnects with a line topology. As this is the case for most interconnects in a circuit, *SBWS* can be applied to them directly. However, there are some interconnects with tree topology. For weighted sink delay objective, those interconnects can be handled by *SBWS* using a similar technique as in [4]. That is we use an iterative algorithm to optimize the tree edges one at a time. At each time we manipulate an edge, we keep all the other edges fixed and apply *SBWS* to that edge. For other objectives like minimizing maximum delay or minimizing area with delay bounds, we can apply the Lagrangian relaxation technique as in [4] to reduce the problems to a problem of minimizing weighted sink delay.

Better theoretical bound on run time: A quadratic run time is proved in Theorem 1. However, the experimental results in Section 4 suggest that the actual run time of *SBWS* is close to linear. In fact, for Lemma 3, we can argue that $\alpha_{i-1} \approx \alpha_i$ and $\beta_{i-1} \approx \beta_i$. This implies that for Lemma 8, if $1/(1+\epsilon/O(m)) \leq R'_n/R_n \leq 1+\epsilon/O(m)$, then $1/(1+\epsilon) < R'_0/R_0 < 1+\epsilon$. So we conjecture that with a tighter analysis, one can prove that *SBWS* runs in $O(n \log \frac{m}{\epsilon})$ time.

References

[1] Chung-Ping Chen, Yao-Wen Chang, and D. F. Wong. Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 405–408, 1996.

[2] Chung-Ping Chen and D. F. Wong. A fast algorithm for optimal wire-sizing under Elmore delay model. In *Proc. IEEE ISCAS*, volume 4, pages 412–415, 1996.

[3] Chung-Ping Chen and D. F. Wong. Optimal wire-sizing function with fringing capacitance consideration. In *Proc. ACM/IEEE Design Automation Conf.*, pages 604–607, 1997.

[4] Chung-Ping Chen, Hai Zhou, and D. F. Wong. Optimal non-uniform wire-sizing under the Elmore delay model. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 38–43, 1996.

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

[6] Chris C. N. Chu and D. F. Wong. A new approach to simultaneous buffer insertion and wire sizing. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 614–621, 1997.

[7] Jason Cong and Lei He. An efficient approach to simultaneous transistor and interconnect sizing. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 181–186, 1996.

[8] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H. Madden. Performance optimization of VLSI interconnect layout. *INTEGRATION, the VLSI Journal*, 21:1–94, 1996.

[9] Jason Cong, Cheng-Kok Koh, and Kwok-Shing Leung. Simultaneous buffer and wire sizing for performance and power optimization. In *Proc. Intl. Symp. on Low Power Electronics and Design*, pages 271–276, August 1996.

[10] Jason Cong and Kwok-Shing Leung. Optimal wiresizing under the distributed Elmore delay model. *IEEE Trans. Computer-Aided Design*, 14(3):321–336, March 1995.

[11] Jim DeTar. Advances outpace SIA roadmap (Semiconductor Industry Association alters projections) (Industry Trend or Event). *Electronic News*, 42(2147):1, December 16 1996.

[12] R. J. Duffin, E. L. Peterson, and C. Zener. *Geometric Programming – Theory and Application.* John Wiley & Sons, Inc., NY, 1967.

[13] W. C. Elmore. The transient response of damped linear network with particular regard to wideband amplifiers. *J. Applied Physics*, 19:55–63, 1948.

[14] J. P. Fishburn. Shaping a VLSI wire to minimize Elmore delay. In *Proc. European Design and Test Conference*, 1997.

[15] John Lillis, Chung-Kuan Cheng, and Ting-Ting Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. *IEEE J. Solid-State Circuits*, 31(3):437–447, March 1996.

[16] N. Menezes, R. Baldick, and L. T. Pileggi. A sequential quadratic programming approach to concurrent gate and wire sizing. In *Proc. IEEE Intl. Conf. on Computer-Aided Design*, pages 144–151, 1995.